Cool Robot Show Report

[ICN5406] Mobile Robot 2018

Student ID: 0745011, 0745012

Name: Petr Gondek, Šárka Weberová

Date: 7/1/2019

1. Purpose:

The purpose of the term project is to create something new and creative. We have a chance to use other supplies, then in the robot contest, such as other types of sensors. It is opportunity to learn how to use some innovative technics and supplies.

We want to create controlling device for mobile robot. The device will be interactable by hands in virtual reality. We want to make it like an entertainment. We will create a virtual panel and using this panel player will be able to control the mobile robot. On the panel, there will be two triggers to control the speed of each wheel.

2. Description of Design:

We use Intel RealSense SR300 sensor, to capture motion of hands. Because of RealSense's minimal system requirements, the camera have to be installed on more powerful device than the RPI3 on the mobile robot. The problem is caused by the USB connector and the CPU. We decided to install the camera on the notebook of one participant.

Game Environment

To develop controlling software, we use Unity 2018. This engine helps us to render game environment. To develop a hand control, we use Legacy Intel RealSense SDK 2016 and its corresponding legacy driver. This SDK can recognize and track a hand in front of RealSense camera, it also contains a kit for Unity which helps to manage the game objects. For making it works we needed to change a little bit Intel's code. The new SDK2 doesn't provide any API for hand detection as the previous ones. So that is why we use the legacy one.

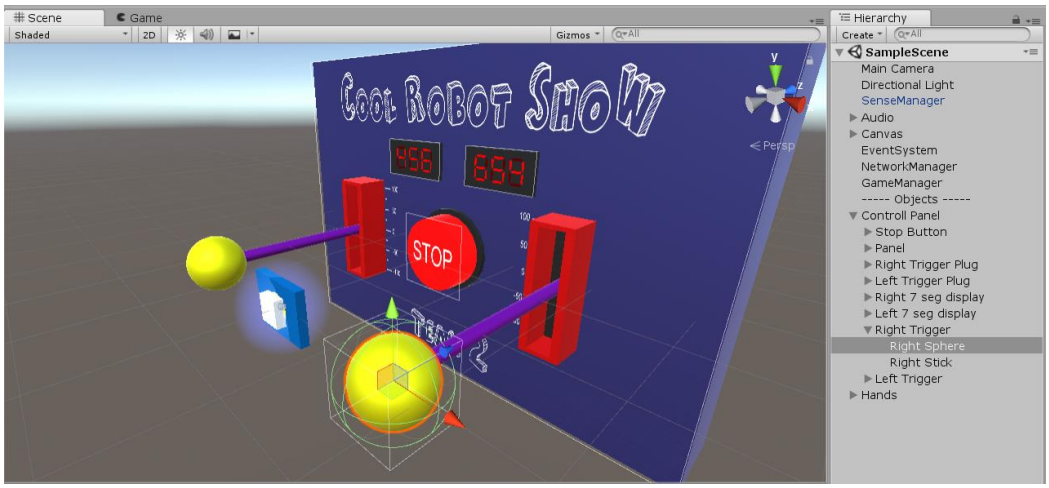
On next figure you can see how easily you can reach the hand data. And like this you can detect most parts of human body. Next code is used to detect closest hand to the Sphere Collider (explained later) and check its gesture.

```
void DetectColesestHand()
{
    PXCMHandData _outputData = senseManager.GetComponent<RSUnityToolkit.SenseToolkitManager>().HandDataOutput;
    PXCMHandData.GestureData _gestureData;
    for (int i = 0; i < _outputData.QueryFiredGesturesNumber(); i++)
    {
        if (_outputData.QueryFiredGestureData(i, out _gestureData) == pxcmStatus.PXCM_STATUS_NO_ERROR)
        {
            if (_gestureData.name.ToString() == "fist")
            {
                PXCMHandData.IHand _hand;
                if (_outputData.QueryHandDataById(_gestureData.handId, out _hand) == pxcmStatus.PXCM_STATUS_NO_ERROR)
                {
                    PXCMHandData.JointData _jointData;
                    _hand.QueryTrackedJoint(PXCMHandData.JointType.JOINT_CENTER, out _jointData);
                    PXCMPoint3DF32 point = _jointData.positionWorld;
                    Vector3 hand = new Vector3(point.x, point.y, point.z);
                    hand.x *= -100;
                    hand.y *= 100;
                    hand.z *= 100;
                    float diff = Vector3.Distance(hand, col.transform.position);
                    if (diff < rangeToDetectHand)
                    {
                        handColId = _hand.QueryUniqueId();
                        sfxTriggerPlayer.PlayGrab();
                    }
                }
            }

        }
    }
}
```

The controlling panel contains two stick triggers to control speed of each wheel. Each stick is able to move in [-30°, 30°]. This angle is sent as percentage to the robot. RPI calculates the real speed for wheel which is sent to Arduino. This is omitted every update so (every frame) so approximately 60 times per second.

To control these triggers, you can use both hands. You need to move palm of your hand to the collision area (green sphere around selected gameobject on next figure) and make a fist gesture. To release trigger, you can move your hand out of the sphere collider or open you hand to send fingers spread gesture.



1 Collider sphere

We created emergency stop button which sends (0, 0) speed signal to stop robot and resets the stick position.

For better user experience we use SFX for grabbing and releasing triggers and pushing the button. When the button is pressed it starts omitting light. Above each trigger you can find 7 segment display which shows actual percentage speed. You can also rotate camera.

Networking

For communication between the robot and the game environment we used sockets. We established network over WiFi between participants. Robot behaves as the server and the game joins to it. We created simple UI for creating the TCP connection.

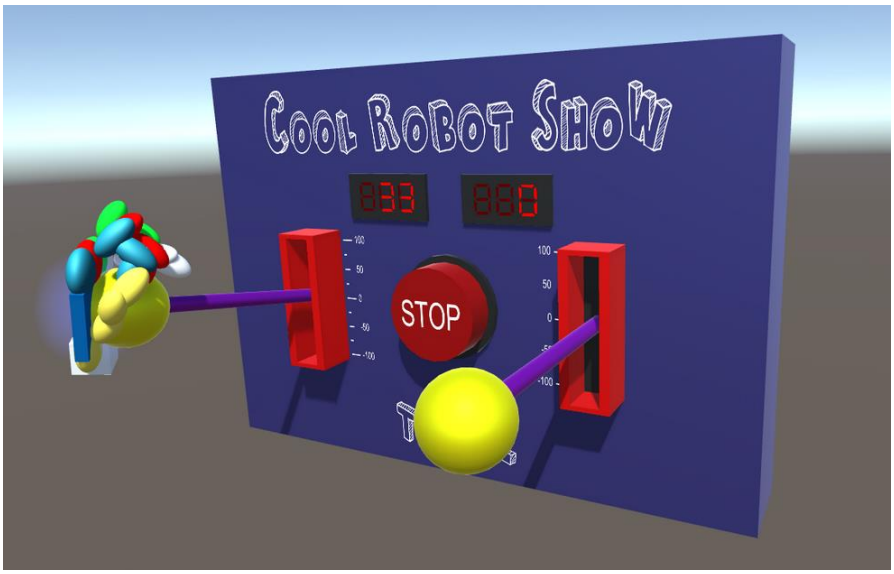
2 Creating connection screen

NetworkingManager in the game environment gathers data from both triggers every frame and sends a message. The message is two bytes long and contains two number, each one-byte long, with speed for each wheel.

3. Result

We successfully created the application as we expected. The PC program was able to recognize the hand in front of the RealSense camera and recognize its gestures. For better usability we also added the button, that stops the robot and puts triggers to neutral (zero) position.

From the spectator's reactions we think that it would be useful to add some mode to be able to move robot in the straight line. It could be done for example by some button. When the button will be pushed, both speed triggers will move synchronously.



3 Running application

4. Discussion

The experience of this task is very exciting. We learned how to create objects in virtual reality and how to work with RealSense camera. We also spent a time by calculating the behavior of virtual components.

Application we have created could be used as a game or to control robot in virtual reality. If another camera will be put on the robot, it could be useful for army purpose, such as field exploration or espionage on the high distances.

Other usages will appear if we put the RealSense camera on the robot instead of the computer. It could be used, for example, on robots in supermarkets. Customers will then have opportunity to control the robot by hand gestures.