KOÇ UNIVERSITY

College of Engineering

COMP 491 – Computer Engineering Design Project

Final Report

LEAP MOTION OPERATOR WITH VR

Participant information:

Mertcan Akardere 34685

Project Advisor(s)
Barış Akgün

06.06.2018

Table of Contents

1. Abstract

Unmanned vehicles, instead of the outdated mechanical controls, could be controlled more intuitively with hand gestures and have more advanced movement commands than simple direction instructions by programming maneuvers therefore increasing efficiency of controls. This project attempted to develop an intuitive, efficient and accurate controls by detecting hand gestures with the help of Leap Motion, and demonstrating this in a simulation with an unmanned vehicle on the Unity, and showing this simulation on a VR headset for the immersive user experience. Hand positions and finger directions were used to control a vehicle while gestures did more advanced commands. These sets of controls were raced against the classic methods to test their effectiveness. Accuracy of the Leap Motion was refined by adding conditionals to counter small losses of input. Thus, new means of controlling a vehicle via hands was developed to challenge the classic methods of control.

Despite unfamiliarity of this method compared to classic methods, it was learnt pretty quickly due to intuitive nature of hand gestures, and advanced maneuvers made possible to challenge efficiency of the older methods regardless of the skill difference between the two, and even though Leap Motion was not precise enough its accuracy was made passable by adding countermeasures in place, thus achieving all around decent quality to be applicable.

2. Introduction

As with the progress of technological advancements, unmanned vehicles have become accessible for commercial and personal use. This is evident with the success of drones, as millions of drones have already been sold to personal use (Gartner 2017)

Currently the means of operating such vehicles are either with the use of provided controllers or in some advanced products with programmable movement. Controllers are precise yet unintuitive.

The purpose of this project is to utilize Leap Motion’s hand tracking ability to operate vehicles in Unity to find alternative methods of controlling unmanned vehicles. Goals to achieve with this control methods were being more intuitive than classic methods, efficient enough to be applicable, and accurate enough to be safe to use for real unmanned vehicles.

There have been similar attempts to this project, where people used Leap Motion to control vehicles like Arduino vehicles (11). But more advanced projects seemed to made use of other devices such as HTC Vive (12). This is probably for tackling the Leap Motion’s inadequate accuracy shortcoming which will be explained further in the following sections. After attempting to deal with these issues with programming and still not achieving satisfying level of accuracy, use of other input devices besides Leap Motion sounds like a more promising study.

Also, JavaScript framework for robotics called Cylon.js has been utilized by Leap Motion developers to access controls of more than 30 platforms via the Internet of Things(IoT), and among them was ARDrone (13). That project had a similar goal, which is developing abstract controls that could be applicable across different vehicle but made this possible making use of a framework that is already in development whereas this project was based on abstract methods written from scratch. Although this project did not use devices on these platforms, supporting this increases scalability significantly as Internet of Things is still getting broader across more devices (14)

3. System Design

First of all, basic physics functions were needed to be added to form the basis for the simulation part. Therefore acceleration, speed and friction were implemented. All of these were separately added both for horizontal speed and rotation speed so that they could be adjusted separately to fine tune a more realistic simulation.

Then functions to perform most basic tasks for controlling a vehicle like going forward, applying brakes and turning were added. These functions would form the abstract base that is called by the advanced movement sets. This abstraction barrier keeps consistency across movement sets but more importantly allows programmed movement sets to be applicable to other vehicles, or even easier transition to controlling unmanned vehicles in real world.

Next step was to implement sensor conditions for the movement sets and methods that call the appropriate basic control tasks. The first movement set was making use of having the hands extended across the Leap Motion, regardless of how the fingers were positioned. There were no defined conditions for just having the hands somewhere on the 3D plane in the API, so two spheres were added to the scene to act as sensors for each hand. These spheres had to borrow a script from the Leap Motion API to be interactable with the Leap Motion's virtual hands that mimic the input data. Then another script to keep track of which hands were in the spheres was added. This scripts output was used with movement set 1's input which called turning, moving forward, or not doing anything depending on the hands. Then it became apparent that a way to apply brakes were required, thus another sphere was added in the center, therefore allowing users to brake by having any or both hands in the middle.

Yet this control method was still failing to achieve the goal of accuracy as there were inconsistencies with the Leap Motion's sensors. This problem was addressed by adding dead zones to moving and not moving the vehicle. Thus, if the vehicle wasn't accelerating, it would take approximately 0.4 seconds of input to move forward to start accelerating which made it less likely to start the vehicle by accident less likely. Also, it would take approximately 0.2 seconds to stop moving forward without the input which dealt with small losses of input data from Leap Motion sensors and allowed for smoother controls.

Next, the movement set 2 methods were implemented, which were moving the vehicle in six directions: moving straight forward, moving forward with turning left or right, moving backwards and moving backwards while turning left or right. Then each of these functions were to be called by a finger direction sensor. Thus, users could control the vehicle by pointing at the direction they wanted to go. These finger direction sensors activation conoids were adjusted so that they were relatively easier to activate but still accurate enough to not cause conflicts. Especially going right were proven to be difficult as the back of the hand could block the Leap Motion's camera to see the fingers.

To further increase the efficiency of movement set 2, a command to change lanes were added. This could be performed by a hand gesture where the user’s index, middle and pinky fingers are extended, and others are not. The direction of the hand decides which lane to change towards. This functionality was added by making use of the extended finger sensor of Leap Motion.

Final method to control the vehicle was controlling it via the arrow keys on the keyboard. This was added to act as a base case for the project so that the results we gathered for the other movement sets were meaningful.

Objects were required to be added to Unity to apply all these codes into. To be visually appealing and depict a more accurate simulation, assets were to be implemented by the Unity’s asset store. These assets were: tocus car for the vehicle, kajaman roads for the roads, text mesh pro for the text on the overlay. Also Leap Motion SDK was added for Leap Motion support and API, Google Cardboard SDK was added to support VR headset output and Trinus VR was added to support Trinus VR app which streams the simulation on the PC to the smartphone that acts as the VR screen. Trinus VR was used as Android SDK was outdated to build the Unity project for android due to version conflicts. Thus, the simulation runs on the PC, then it is streamed to a smartphone via USB cable or the internet, then the smartphone acts as a screen for a cardboard VR headset.

Final step was turning the roads into a test course. This was done via adding checkpoint lines on the road. Time it took to pass through each checkpoint was recorded to get quantitative results on the efficiency of the control methods.

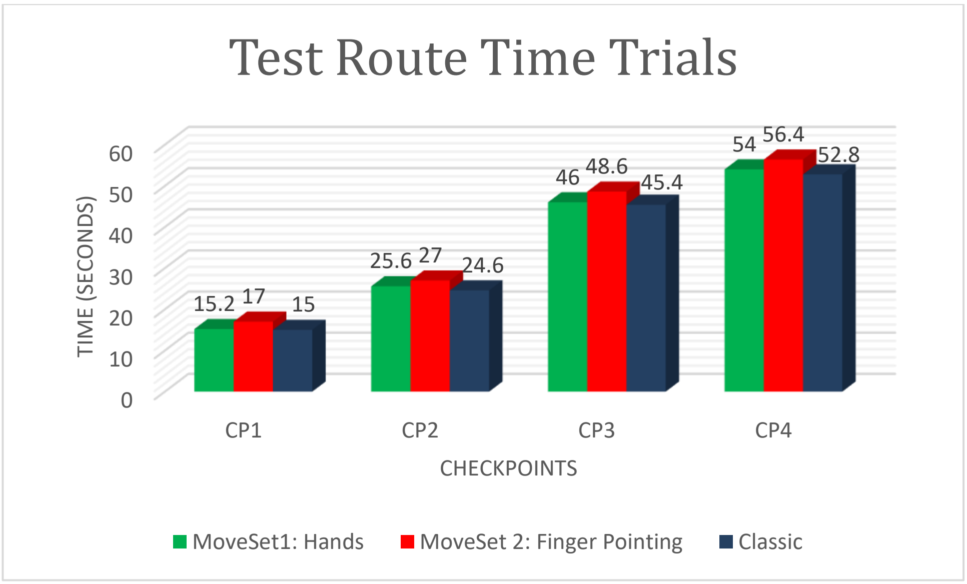
4. Analysis and Results

Among the intuitiveness, efficiency and accuracy goals of the control method, only the efficiency was possible to put into numbers. To compare efficiency across the movement sets, the road was turned into test course with checkpoints, and time for multiple attempts were recorded for each movement sets (Table 1).

Table 1: Time Trial for each Move Set on the 4 checkpoints with 5 attempts

MS1	T1	T2	T3	T4	T5	**AVGs**
CP1	15	14	14	16	17	15.2
CP2	24	26	25	26	27	25.6
CP3	46	46	45	46	47	46
CP4	54	54	53	54	55	54
MS2						
CP1:	17	16	20	17	15	17
CP2:	27	26	30	27	25	27
CP3:	47	46	54	50	46	48.6
CP4:	55	54	61	58	54	56.4
Classic						
CP1:	15	15	15	15	15	15
CP2:	24	24	25	25	25	24.6
CP3:	47	46	45	44	45	45.4
CP4:	53	53	53	52	53	52.8

Table 2: Averages for each checkpoint are compared for each Move Set (shorter the better)



The classic control method outperformed the Leap Motion ones (Table 2), but this was not surprising due to experience difference. The test subject, me, used keyboard to move vehicles for more than hundreds of hours but my experience with Leap Motion was less than few hours. More interestingly, the first movement set outperformed the second movement set even though first one was much simpler. This was due to accuracy issues of Leap Motion, like difficulty to turn right in the second movement set.

Personal observations for the accuracy of Leap Motion’s control methods were unfortunately not promising, therefore the accuracy of the movement sets was also inadequate to control real vehicles despite the various countermeasures implemented. The faulty input from Leap Motion was received when the hands were in a position that was blocking the view of the fingers, or when one of the hand suddenly left the screen, or when user’s sleeves were long, or it was hard to detect if it was the palm or the back of the hand for the camera. Thus, there were too many conditions were inaccuracies occurred, and the best way to regain control was to reset the hands by showing the palms of both hands across the camera.

Personal observations for the intuitiveness of the movement sets were made during the demo of the project as many people had to experience Leap Motion and the movements sets for the first time. Even though using hand gestures should be more comfortable than using electronic devices, most of the people were seeming to be intimidated by making mistakes which I believe is a result of familiarity with electronic devices using similar controls while Leap Motion is quite different. Yet it usually took less than minute for people to understand how to use the vehicles. They needed more time to adapt the Leap Motion’s inaccuracies, but they quickly adapting to be more effective drivers. Thus, even though it seemed intimidating by its unfamiliarity, people were quick to learn it which suggests it is quite intuitive.

5. Conclusion

Aim of this project was to utilize Leap Motion to control a vehicle, and this aim was achieved with two different movement sets. These movement sets success rate was determined by their effectiveness, accuracy and intuitiveness.

Due to experience difference and quicker controls, the classic methods performed better than the movement sets. Yet the difference was no great, indicating that the effectiveness of the movement sets was passable.

Despite being intimidating at first due to its unfamiliar design, using Leap Motion was being learnt at a considerable pace, suggesting its decent intuitiveness.

Even though there were countermeasures for the inaccuracies that may occur with Leap Motion were implemented, they were not enough to get a solid input which reduced effectiveness, increased frustration and confusion among new users that did not know how to deal with the issues and increased the chance of accidents. Despite being usable in virtual world with reset button, this would not be consistent enough to use an unmanned vehicle in real world.

Simulation and sensor sensitivities in Unity could be quick improvements on the project. But the main issue that was holding the project back was the inaccuracy of the Leap Motion sensors. Main issue with the sensors were losing track of the hands and fingers due to obstructions, so Leap Motion can be improved by another camera across the user. Alternatively, another device can be used to gather hand position and gestures.

Consequently, using hand gestures to control a vehicle seemed plausible and promising, yet the Leap Motion’s accuracy proved to be inadequate for using it on unmanned vehicle in the real world. Next steps to improve this project would be adding support for other input devices like HTC Vive to get more precise controls and add support for Cylon.js to operate robots on more than 40 different platforms.

6. References

Versions of the project can be accessed through:

www.github.com/MertcanAkardere/Leap-Motion-Operator-with-VR/branches

1) Leap Motion, 2017, www.leapmotion.com

2) “Gartner Says Almost 3 Million Personal and Commercial Drones Will Be Shipped in 2017.” Gartner, 9 Feb. 2016 www.gartner.com/newsroom/id/3602317

3) Stansbury, Richard, et al “Ethical Concerns of Unmanned and Autonomous Systems in Engineering Programs.” 121st Annual ASEE Annual Conference & Exposition, June 2014.

4) Maltbie, Benjamin, “Powerful Illusions: Addressing the Code of Ethics for VR.” 16 Apr. 2017 www.uploadvr.com/ethics-vr-research-concerns/

5) Rolf Molich, Brenda Laurel, Carolyn Snyder, Whitney Quesenbery, and Chauncey E. Wilson. 2001. Ethics in HCI. In CHI '01 Extended Abstracts on Human Factors in Computing Systems (CHI EA '01). ACM, New York, NY, USA, 217-218. DOI=http://dx.doi.org/10.1145/634067.634197

6) S.Richmond, 14 Aug. 2013, “How to use Git for Unity3D source control?”, www.stackoverflow.com/questions/18225126/

7) Google VR, www.developers.google.com/vr/

8) Trinus Virtual Reality, www.trinusvirtualreality.com/

9) 3D Low Poly Car For Games (Tocus), assetstore.unity.com/packages/3d/vehicles/land/3d-low-poly-car-for-games-tocus-101652

10) Kajaman’s Roads – Free assetstore.unity.com/packages/3d/environments/roadways/kajaman-s-roads-free-52628

11) KIVI (Leap Motion Controlled Arduino Vehicle), 27 Jan. 2018, www.instructables.com/id/KIVI/

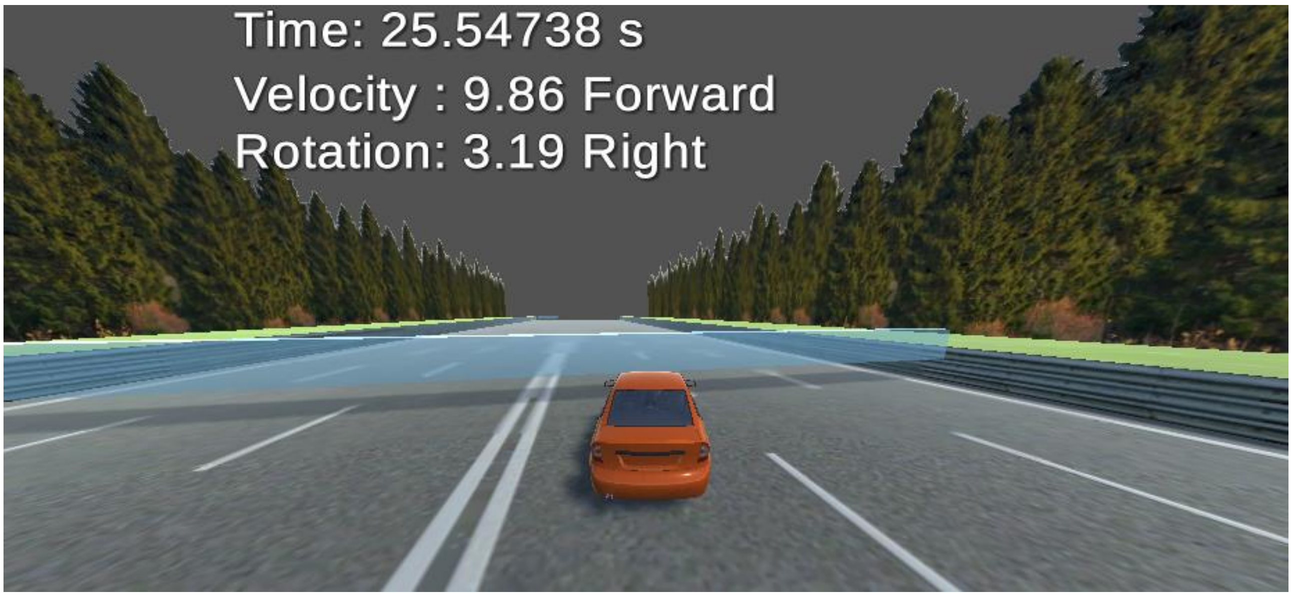
12) Leap Motion and VR Car, 28 Feb. 2017, www.youtube.com/watch?v=gKG4FpLnjzk

13) Control Robots with Cylon.js, 12 May 2015, blog.leapmotion.com/featured-platform-control-robots-cylon-js/

14) Cylon.js – JavaScript framework for robotics, www.cylonjs.com/

7. Appendix

Image 1: Simulation in action



Time: 25.54738 s

Velocity : 9.86 Forward

Rotation: 3.19 Right

Image 2: VR headset with the Leap Motion attached

Code 1: Dead Zone code that introduces minimum input time for both hand sensors in movement set 1

```
    private void deadzoneCheck(bool isLeft)
    {
        if(isLeft)
        {
            if (!LeftHand)
            {
                if (dzLeft == 0)
                {
                    enterLeft();
                    dzLeft = dzMove;
                }
                else
                    dzLeft--;
            }
            else if (dzLeft < dzMove)
                dzLeft++;
        }
        else
        {
            if (!RightHand)
            {
                if (dzRight == 0)
                {
                    enterRight();
                    dzRight = dzMove;
                }
                else
                    dzRight--;
            }
            else if (dzRight < dzMove)
                dzRight++;
        }

            }
```

Code 2: Input Decay code that turns sensors off only after receiving no input for a short while

```
    private void inputDecay()
    {
        if (brakeText.color.a > 0)
            brakeText.color = new Color(255, 0, 0,
brakeText.color.a - 1);

        if (LeftHand)
            dzLeft--;
        else if (dzLeft < dzStop)
            dzLeft++;

        if (RightHand)
            dzRight--;
        else if (dzRight < dzStop)
            dzRight++;

        if (dzLeft < 0)
        {
            exitLeft();
            dzLeft = dzStop;
        }
        if (dzRight < 0)
        {
            exitRight();
            dzRight = dzStop;
        }

            }
```