FINAL REPORT

HARDIS

with

LEAP MOTION

OPENROV

06/06/2016

Report for CSE 145 – Spring 2016

This report will present our solution to use the Leap Motion Module as a controller for any robotic device, in particular the OpenROV submarine drone.

The three students working on this project are: Antonino Fugazzotto, Guillaume Hauss, and Yuming Qiao, with the help of Jim Trezzo and Professor Ryan Kastner.

Table of Contents

THE IDEA 2

FIRST THOUGHTS 3
- About the device 3
- About the tracking 3

HARDWARE 4
- Leap Motion 4
- OpenROV 5

SOFTWARE 5
- Leap Motion tracking program 6
- OpenROV plugin 6

PROJECT MANAGEMENT 7
- Schedule 7
- Team work 8

APPENDIX 8

BIBLIOGRAPHY 8

Final Report

REPORT FOR CSE 145 – SPRING 2016

THE IDEA

What separates mankind from the animal reign is the ability to make complex tools. Those tools enhanced our way of life: first it was simple torch, then walls, then farming tools to have sustainable food, and we kept developing new tools until someone invented the computer. The device you have in front of you enables you to read this paper wherever you are, whenever you want. It does amazing things.

But so does a piano, or a flying drone, or a robotic assistant. With each of these devices you can interact with your environment: create music, discover the sky, help someone… The only requisite is to have the capacity to control these devices. How to play the piano without any piano? How to make the drone fly if you don't have or see the remote? How to call for help if you can only move 3 fingers, in case of disabled people?

This is where our project finds its purpose. Hardis® is a way to enable your fingers to do more. It is a way to allow you to interact with things that you can't normally interact with, in specific contexts. Try to play the piano in a plane! You'll tell me how this went! With Hardis® and a headphone, you can practice every instrument you want, while seating in a plane. Try typing on a keyboard when wearing an Oculus Rift®, I bet you won't be able to write your name correctly.

Hardis® tracks both of your hands, meaning your two palms and your ten fingers, with an extreme precision. Our software will analyze YOUR moves and, according to a database YOU would have created, it will generate the command YOU need to perform. The next step is to connect to the device you want to control: a piano, a drone, a robot. The final step is to educate the device, in order to make it do the right move when receiving the command you generated. And that's it! Three steps, and you have a piano in your pocket. Amazing, huh?

FIRST THOUGHTS

About the device

As mentioned before, our system needs a device to play with. Hardis® is simply a universal remote, fitting in your pocket. Our first idea was to build a very simple ground robot. Three wheels, a platform to receive the hardware (communication, control, actuators) and we're good to go. It clearly appeared that thinking of this ground robot was not the same challenge than creating it. Ten weeks is not a long time, and we needed to focus on the tracking system. So we moved on another solution.

There were many iRobots at our disposal, so we started thinking about using one. The ground platform was already functioning. We just needed to build the control and communication systems. Again, on the paper, that seemed really easy. A Bluetooth module to make the remote talk to the iRobot, an Arduino to translate the commands received into real movements, the concept was easy. However, that implied that we had to look at the iRobot's API, which really lacked of precision and updates. Time was our enemy, and we couldn't afford to lose weeks trying to figure out how to move the robot forward. iRobot was no longer an option.

Fate then came into play in the person of Jim TREZZO. He is working on an OpenROV (a submarine drone, which will be detailed later in this report), and presented us this open-source worldwide-community project. This drone met all the qualifications we were looking for. The communication system was already in place, using a socket.io channel (also detailed later on); the command system has been fully developed and tested by thousands of people, meaning the API was precise and easy to use. We found our device to play with.

About the tracking

Tracking the hands is the core technique of our system. We needed something precise, fast, small, and easily compatible with any hardware platform. Therefore, we immediately rejected systems such as Matlab, FPGAs, and Arduino to perform the tracking.

Lucky for us, Leap Motion exists. Again, this device fit perfectly our needs: an image sensor the size of a lighter, pluggable to any computer via USB, with an API in more languages than we can even imagine. The choice was made and we purchased it in the second week, in order to start working on it the soonest possible. After a quick hands-on session, we became aware that a lot of data was available while tracking the hand: fingertips position, palm's Euler angle, palm's normal and direction vectors, grasping strength, arm orientation, etc.

We first thought about tracking the fingertips positions: huge possibilities therefore a lot of available commands. However, the data was not stable enough to perform a fast matching with the recorded gestures. That is easily explainable by the fact that you'll never place your fingers at the exact same locations twice in a row, even though you want to. Thus, we forget about absolute data, such as location, and focused on relative data, such as angles and orientation. These parameters were less sensitive to hand's instability and implied a more natural movement of the hand: tilting according three axes is basically the purpose of a hand. We agreed on that idea and focused on three values: pitch angle, yaw angle and grasping strength. These values are bounded and easy to fetch from the Leap Motion SDK.

HARDWARE

Leap Motion

Leap Motion moto is to remove barriers between people and technology. They believe in a simple, natural input as a key to unlock any technology. Your hands are such a simple input.

Here is the image sensor. It's the size of a lighter, with a single USB cable plugged on one side. The green led is on the side facing the person. Following the rule of the direct frame, the x and y axis are defined as shown in the picture

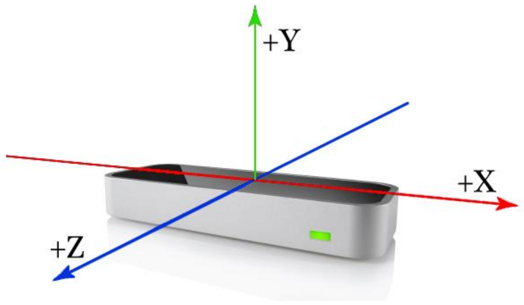

Figure 1 : Leap Motion axis

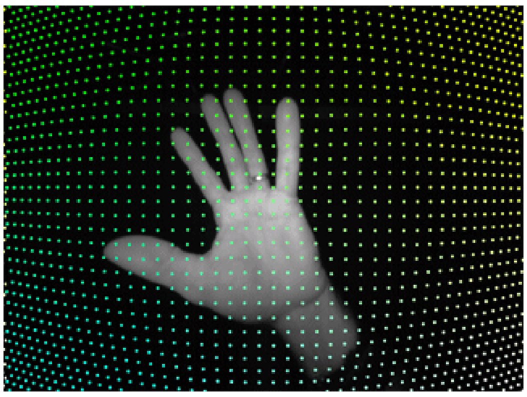

Figure 2 : Leap Motion Image

Three infrared LEDs are projecting a point grid, and an infrared camera is capturing images of objects and the grid. Some complex image processing algorithms are then run to fetch the position of any object inside the viewing space.

The hand is then rendered if needed. Here are shown two important vector:

- The normal vector of the hand's palm
- The direction vector of the hand's palm

These two vectors define a plan that is used to calculate Euler angles of the hand.

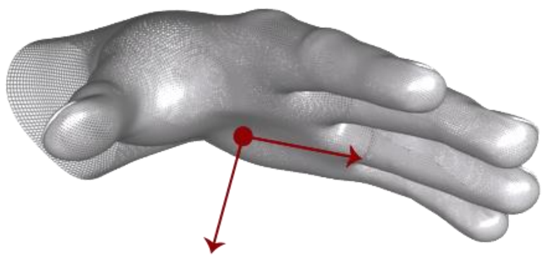

Figure 3 : hand vectors

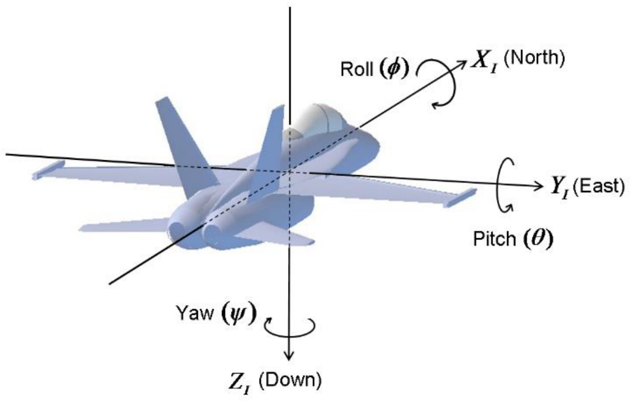

Figure 4: Euler angles

As mentioned before, we are using pitch and yaw angles to control the OpenROV. If you are not familiar with such parameters, this graph will explain what these angles are.

OpenROV

OpenROV is a worldwide-community open-source project focused on designing the most modular, easy-to-build and cheapest submarine exploration drone. Hardware and software are constantly upgrading, allowing each and every one of us who wants to work on that project, to develop new abilities or functionalities. The core idea is that you can build this drone with on-the-shelf materials and a little bit of brain.

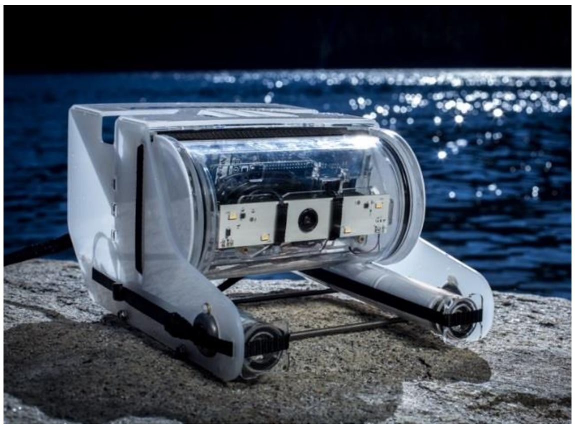

Figure 5 : OpenROV in the field

The drone is tethered to a computer (wireless communication underwater is not that easy to use) with a 100 meter Ethernet cable, allowing real-time communication and feedback. The power is supplied by 6 batteries on the OpenROV, used by flash lights, camera and propellers. As to now, the drone is purely exploration focused. No robot arm has been developed to interact with its environment. We can imagine that Hardis® could be used in such context, where the robot arm will repeat the gesture of the man's hand. That's the beauty of our system: its adaptability is nearly infinite.

Deepen the understanding of the propulsion system of the OpenROV does matter because it is our target. Three propellers allow the drone to fly undersea. One top propeller is used to lift or sink the drone, two back propellers are responsible for thrust and yawing. The power of these propellers are stepped onto 5 levels, and they are binary activated once they need to. To put this in other words, the robot can speed up and turn right in the same time, for example. Direction is decoupled from speed.

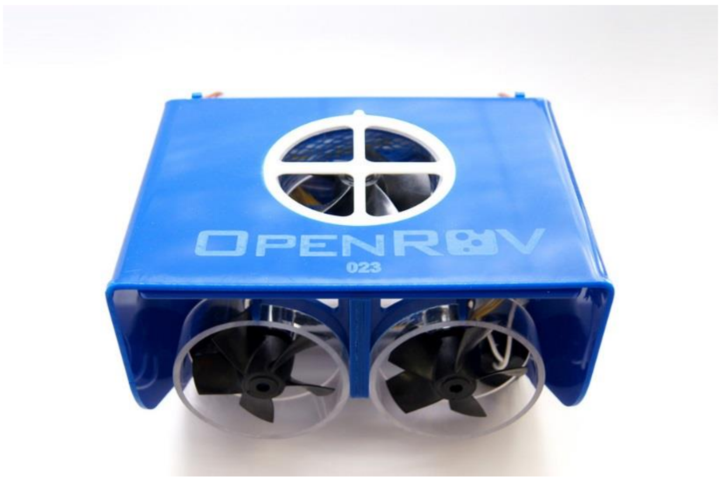

Figure 6 : Propellers

Figure 7 : Flash lights and camera

As mentioned before, the OpenROV is equipped by a camera and two flash lights. In further developments of our system, it would be easy to add new gestures recognition matching new commands, such as taking a picture or switching on the lights.

SOFTWARE

Leap Motion tracking program

As mentioned before our initial input is your hands' gestures. The Leap Motion module is there to capture these gestures, and extract the physical variables used to generate the OpenROV commands. We focused on the pitch angle, the yaw angle and the grasping strength to determine the movement we want the OpenROV to perform. Basically, this drone can move forwards, backwards, turn right, turn left, go up and go down: six moves matching six possible binary combinations of these parameters.

The first step is to continuously get the frames from the Leap Motion sensor. This loop does that for us:

```
Leap.loop(controllerOptions, function (frame) {...});
```

This variable counts the number of hands perceived by the Leap Motion. Different functions will be called depending on its value.

```
var handsCount = frame.hands.length;
```

Then we focus on the right hand to begin with, using this variable:

```
var hand = frame.hands[0];
```

To assure a proper interaction, the hand needs to be face down. We check that using this code.

```
var normal = hand.palmNormal;
var normX = normal[0];
var normZ = normal[2];

if ((normX < -10 || normX > 10) && (normZ < -10 || normZ > 10)) {...}
```

Finally, we capture the interesting parameters using:

```
var pitchInput = hand.pitch();
var yawInput = hand.yaw();
var thrust = hand.grabStrength();
```

The Leap Motion API is really helpful here. The pitch and yaw values are bounded by -1 and 1, the grab strength by 0 and 1. Matching parameters values with binary commands is then really easy.

OpenROV plugin

The software architecture of the OpenROV cockpit allows a very easy plugin addition. Therefore, we decided to go on this path, instead of designing a whole new software, that would have been interacting with the OpenROV.

First, we will explain how the OpenROV is controlled, and then our plugin will be detailed.

At any time, the state of the OpenROV, in movement or at rest, is defined by this JSON object:

```
self.positions = {
       throttle: VALUE_1, yaw: VALUE_2, lift: VALUE_3,
       pitch: VALUE_4, roll: VALUE_5, strafe: VALUE_6
};
```

The throttle value is responsible for the speed of the OpenROV, the yaw, pitch and roll values defined the orientation of the drone and the strafe value is the camera control command. To set these values, we call these functions:

Move forward:

```
rov.setThrottle(1);
```

Move backward:

```
rov.setThrottle(-1);
```

Turn right:

```
rov.setYaw(1);
```

Turn left:

```
rov.setYaw(-1);
```

Move up:

```
rov.setLift(-1);
```

Move down:

```
rov.setLift(1);
```

Our task is then to call these functions at the right time, given the Leap Motion gesture input. We use simple if-loops to perform this setting step. Every new frame provided by the Leap Motion gives a new set of data, meaning that we can set commands values with a high rate and accuracy.

PROJECT MANAGEMENT

Schedule

We had to come up with a schedule and a list of provable milestones, to make sure that we were on the right tracks. Professor Kastner advised us to specify one milestone per person per week. This precision guarantees that everyone is involved in the project at any time, and that each aspect of our system is equally developed. Here is our schedule.

Member	Week 1	Week 2	Week 3	Week 4	Week 5
Guillaume	Set the project scope	Define the final goal	Pitch the project	CRUD program to manage motions used	Find the commands to send to the OpenROV
Yuming	Join the team!	Find the tracking system	Get data from LeapMotion	Find the right data to use from the LeapMotion	Architecture of the OpenROV plugin
Tony	Join the team!	Design the system architecture	Simple LeapMotion Application	Demo of OpenROV plugin (50%)	Demo of OpenROV plugin (100%)

Member	Week 6	Week 7	Week 8	Week 9	Week 10
Guillaume	Final report (50%)	50% Video script and icons for the Oculus Rift	100% Video Script	Final report (100%)	Video making
Yuming	JS program to get these data	25% plugin code	50% Get the stream live video	100% Get the stream live video	JS program documentation
Tony	Architecture of new OpenROV plugin	25% plugin code	75% plugin code	100% plugin code	JS program documentation

Team work

We decided not to have a team lead, because there was no need for one. Being only three students working on that project, making decisions were not that difficult and tasks division was naturally done based on one's preferences. Indeed, we were lucky to have complementary competences, therefore each one of us focused on the aspect he was more comfortable with. We had weekly meetings to report the progress on our assigned task. We also used this time to change our core idea (Database/no database, external software/internal plugin, Java/JavaScript…) Therefore, our project is perfectly adaptable to the OpenROV exploration context.

APPENDIX

Leap Motion: www.leapmotion.com

OpenROV: www.openrov.com

Plugin LeapMotion: *https://bitbucket.org/afugcse/hardis_cse145_sp16.git*

BIBLIOGRAPHY

Figures 1, 2, 3:
https://developer.leapmotion.com/documentation/javascript/devguide/Leap_Overview.html

Figure 4 : http://www.chrobotics.com/library/understanding-euler-angles

Figure 5: http://www.openrov.com/products/2-7.html

Figure 6: http://www.thehulltruth.com/marine-electronics-forum/651823-openrov-remote-controlled-underwater-drone-1-000-a.html

Figure 7: http://www.3drc.es/sherlock-mi-openrov/