

嵌入式系統技術實驗 - Final Project

第23組 0710121 陳泳翰
0612020 吳峻陞

Topic:

使用2D光達(LiDAR)實作3D空間建模

Inspiration:

在Apple新推出的iPhone 12中，新增在鏡頭組中的LiDAR使iPhone可以透過環境測距搭機虛擬場景選擇最適合的拍照演算法，模糊化、邊緣強化、人像辨識都因此而受益，也是iPhone在硬體條件下不如其他安卓旗艦手機卻仍有本錢和他們互相競爭。

除了對圖片演算法有正向幫助以外，LiDAR的環境搭建在土木工程上也有相當的益處，傳統的土木工程需要利用精準測距儀和額外人力成本將空間環境精準地確定下來，在現今的科技發展下，LiDAR擁有與精準儀器相去不遠的距離量測，但LiDAR可以省下巨量的時間成本和人力，所以使用LiDAR執行3D空間建模是一個非常promising的項目。

透過LiDAR建構出的3D空間相對於人工測量更可以有應用的潛力。在土木安全上，可以利用建構出的3D空間判斷出環境的不良處，例如梁柱的損壞、地基的穩定等等；在救災安全上，坍塌的建築物大大的增加了救災的難度，以往傷者的尋找都必須要有人深入災場才有辦法，但這往往使救災者曝露在危險底下，利用LiDAR的空間建模就可以減少人員的傷亡，並且透過建模出的空間快速鎖定有危險的區域和定位傷亡者，讓救災更有效率；除了環境以外，3D空間建模可以幫助3D列印製造模型，將物體以LiDAR掃描後就能夠更簡單的複製出想要的物體。

基於上述的優點，我們想要利用市售的2D測距LiDAR和自轉軸搭建出一個可以進行3D空間模擬的儀器，讓3D模擬不只侷限在貴重儀器範疇和實驗室。

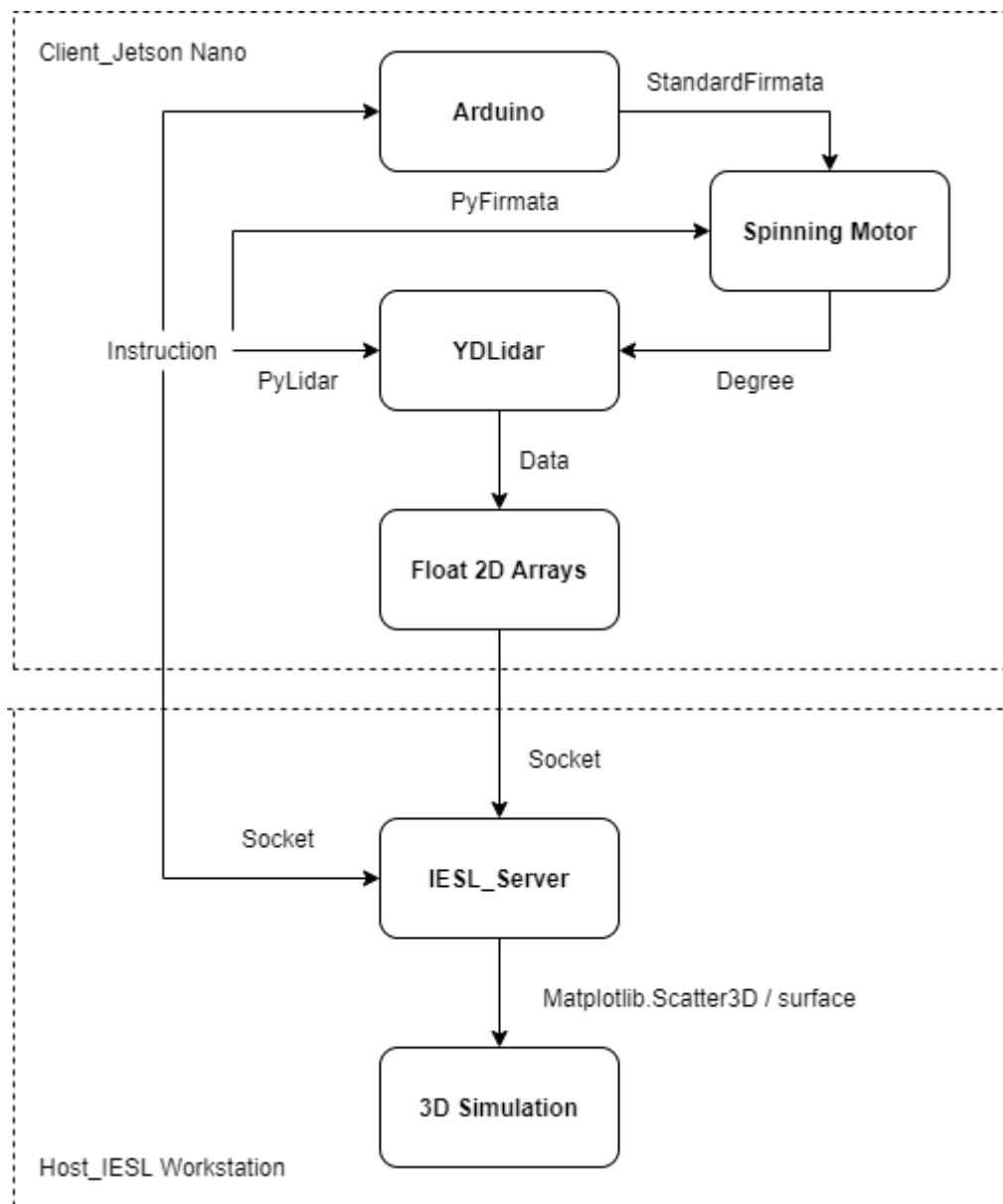
Concept:

利用2D LiDAR配合180度自轉馬達建構出一個360度 \times 180度的球體環積分，LiDAR以垂直的方式掃描360度的平面圓(r, θ)，自轉馬達則是球座標的水平 ϕ ($0 \sim 180$)角，利用3個維度空間(r, θ, ϕ)的補綴把平面掃描圖形在各個角度堆疊起來，如此一來就能夠搭建出一個完整的3D球座標，最後利用matplotlib把座標標示並連接起來即可得到3D的空間建模。

File Description:

1. **scan.py** - 主要執行檔，裡面包含了host(IESL_Workstation) <-> Client (Jetson Nano)連接、LiDAR掃描執行和最佳化、維度矩陣轉換和切割、自轉馬達與Arduino的連接和角度控制、還有3D建模資料的preprocessing。
2. **server.py** - 在工作站接收資料並執行3D建模的python檔，在運行後等待接收Jetson Nano傳來的資料，於資料傳送完成後建構3D空間並顯示於螢幕。
3. **pylidar.py** - 開源資料，使LiDAR可以執行並儲存距離為360度的dict。
4. **usb_tool.py** - 確定Jetson Nano上面port使用狀況，於開啟端口時使用。
5. **StandardFirmata.ino** - Arduino內建程式，使Jetson Nano可以透過python控制自轉馬達。

Structure & Flow:



首先執行server.py使工作站處於oncall狀態，等待client(Jetson Nano)資料傳入，之後操作Jetson Nano，這邊Instruction泛指在Jetson Nano上以scan.py發出的指令，載了StandardFirmata.ino的Arduino受PyFirmata控制而讓自轉馬達轉動後，Instruction利用PyLidar控制YDLIDAR執行掃描，掃描後的dict轉置為三個2D的矩陣，矩陣在scan.py內進行優化處理並以.txt檔案的方式儲存，這邊 .txt檔案可以幫助我們把lost的點以零的方式補起來，而非NaN，讀取儲存的矩陣後將之進行拆解(因socket有傳輸限制)，把拆解後的矩陣送入host，host一接收到資料就會利用matplotlib3D將空間建模出來。

Hardware & Environment:

硬體的部分分為三個: 1. Arduino與自轉馬達 & 2. Jetson Nano 2G & 3. LDAR

1. Arduino連接的port分別有5V、GND和port 9, port 9是控制角度的端口, 控制自轉馬達每次轉動的 ϕ 角, 以本次的實作為例, 我們每次轉動的 ϕ 角為1度, 總共旋轉180次, 使用的函數為StandardFirmata, 讓Arduino可以接收從Jetson Nano來的訊號(角度), 當Jetson Nano使用PyFirmata傳送角度至Arduino時, Arduino可以被動的傳送指令給自轉馬達轉到想要的角度。

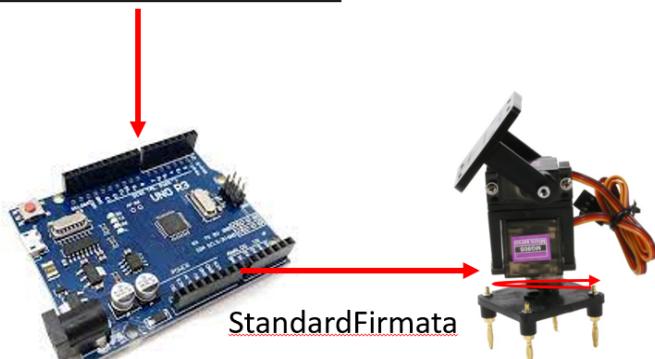


Arduino



自轉馬達 MG90S

```
116 #connect to Arduino (Motor)
117 port = "/dev/ttyACM0"
118 pin = 9
119 board = Arduino(port)
120 board.digital[pin].mode = SERVO
121 board.digital[pin].write(i)
```



自轉馬達控制示意圖

2. Jetson Nano是此次實作的主角, 執行的python檔案會在此開發板上面進行, Jetson Nano的OS為Ubuntu 18.04, apt和apt-get等相關套件皆更新至最新, python3版本3.7.3, 右下圖為我們有使用的函式庫, 皆是在18.04下的最新版本。



Jetson Nano 2G

```
1 import pylidar
2 import matplotlib.pyplot as plt
3 from mpl_toolkits.mplot3d import Axes3D
4 from time import sleep
5 import timeit
6 import numpy as np
7 from pyfirmata import Arduino, SERVO
8 import socket
```

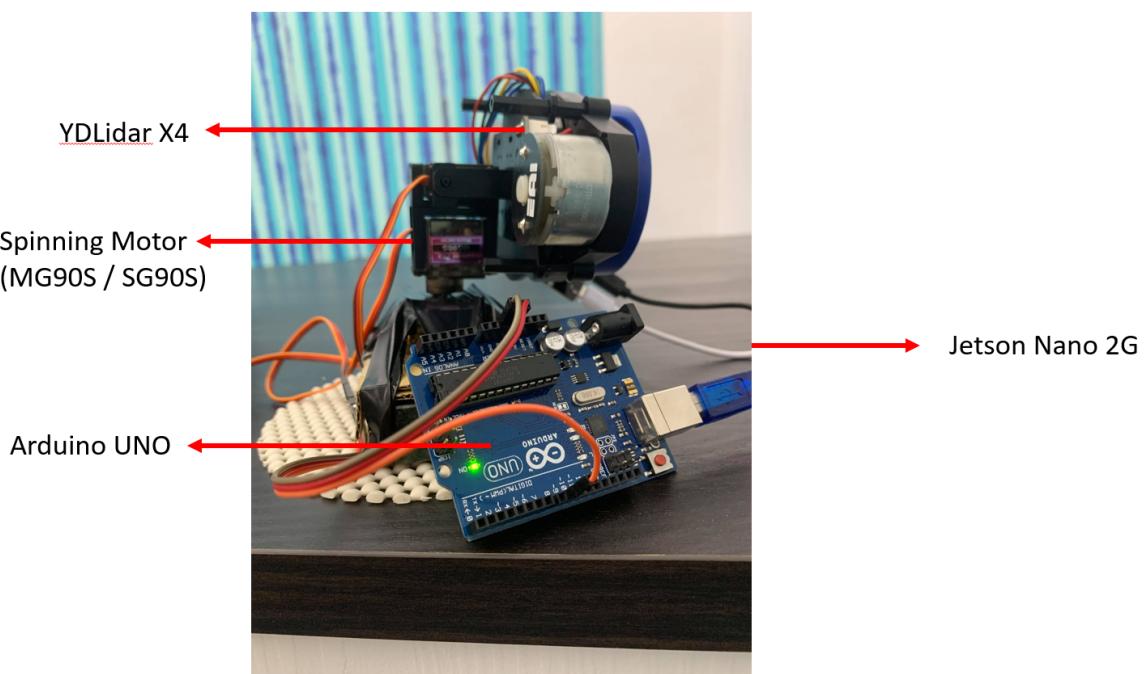
Libraries from Python

3. LiDAR使用YDLiDAR X4，他可以以360度掃描平面距離並將距離存為dict以供後續使用。



YDLiDAR X4

將全部硬體部分搭建成之後如下圖，Arduino利用USB cable連線至Jetson nano並用排線和自轉馬達連接，LiDAR也以USB連接至Jetson Nano上，而自轉馬達和LiDAR在雲台的平面上使用膠帶固定，防止轉動時的大幅度傾斜和錯位。



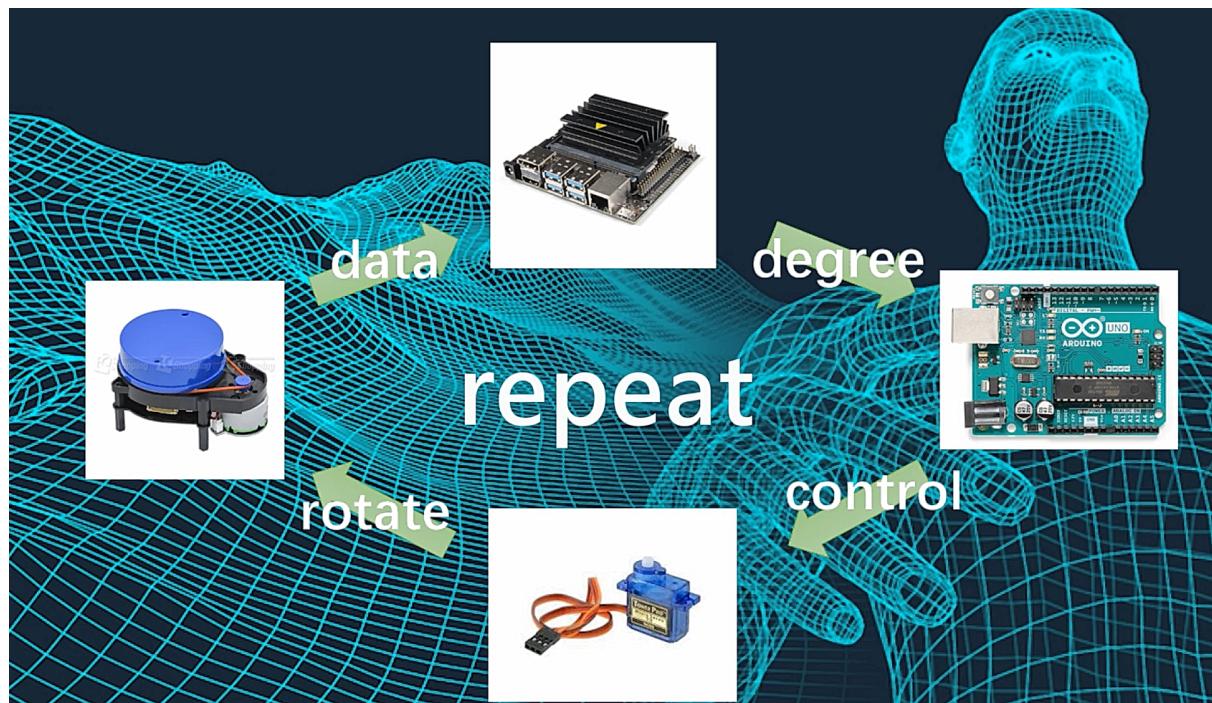
Algorithm & Implementation Method:

1. 我們利用YDLiDAR x4收集數據到python平台，因此引進官方函式庫PyLidar。在設計演算法前原本只有要先看資料回傳形式，但在初步操作時發現好像因為產品瑕疵所以無法順利收集數據，因此又研究了操作LiDAR的各種指令才讓他順利運作。

2. 檢查回傳的資料型態我們得知，LiDAR會不斷回傳資料(θ , distant)，而PyLidar函式庫是每一次迴圈都接收360筆資料，將 θ 取floor然後統計各角度(0-359)收集到的數據取distant平均值，最後回傳一個dict，這樣的收集方式讓我們有以下問題和決定：

- 是否該增加theta精確度和取值數量？但在考量硬體算力和嘗試取720種角度之後覺得目前有點多餘，所以維持原本的預設精確度。
- 因為每次回傳不一定每個角度都有數據，所以該怎麼有效率地補上沒有的distant數據？有嘗試過線性差值及鄰近差值，但最後效果都沒有多掃2-3圈好，所以最後還是多掃幾次盡量減少缺失的角度，再取平均或在小範圍差值。
- 怎麼取得各點座標？因為LiDAR掃描2D，且舵機馬達最多轉180度，所以聯想到球座標，但受限於掃描擺放方式所以計算完所有座標值後在對調yz軸。作法是預先使用單位球體為基準分別設置好xyz座標矩陣，各矩陣都是360*180的二維矩陣，最後在對應的矩陣位置乘上distant，描繪出整個空間外貌。

3. 訊號傳遞：



這是我們的signal flow, 主要是以下重複：

- LiDAR掃描並傳遞資料給Jetson Nano
- Jetson Nano要求Arduino轉換角度
- Arduino轉動舵機馬達 (齒輪切換需要時間，所以有額外使用sleep等待角度穩定)
- 舵機馬達帶動LiDAR

4. 最後因為Jetson Nano圖形算力有限，因此選擇先將計算好的座標儲存下來再用socket回傳到iesl工作站，但socket傳輸有大小上限且需要時間，所以我們降低浮點數精確度(float64 -> float16)，並把360*180拆成180個一為陣列一一傳輸，但後來想到，其實我們不用急著把座標算出來，可以傳distant過去就好然後。在server上計算座標並模擬即可。

5. 以下是code在各階段實現:

Host <-> Client Passing

```
166     for i in range(0,180):
167         data = s.recv(2048)
168         s.sendall(xx[:,i].tobytes())
169         data = s.recv(2048)
170         s.sendall(yy[:,i].tobytes())
171         data = s.recv(2048)
172         s.sendall(zz[:,i].tobytes())
173         data = s.recv(2048)
```

Host <-> Client Receiving

```
20     for i in range(0,180):
21         clnt.send(b'\x00')
22         data = clnt.recv(2048)
23         x.append(np.frombuffer(data, dtype='float16'))
24         clnt.send(b'\x00')
25         data = clnt.recv(2048)
26         y.append(np.frombuffer(data, dtype='float16'))
27         clnt.send(b'\x00')
28         data = clnt.recv(2048)
29         z.append(np.frombuffer(data, dtype='float16'))
```

Connections of LiDAR, Motor, and Host

```
104
105 #connect to host to plot
106 HOST = '140.113.217.202'
107 PORT = 6011
108 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
109 s.connect((HOST,PORT))
110 print("HOST Connect!")
111
112 #connect to Arduino (Motor)
113 port = "/dev/ttyACM0"
114 pin = 9
115 board = Arduino(port)
116 board.digital[pin].mode = SERVO
117 board.digital[pin].write(i)
118
119 #connect to YDLidar
120 port = "/dev/ttyUSB0"
121 obj = pylidar.YdLidarX4(port)
122
```

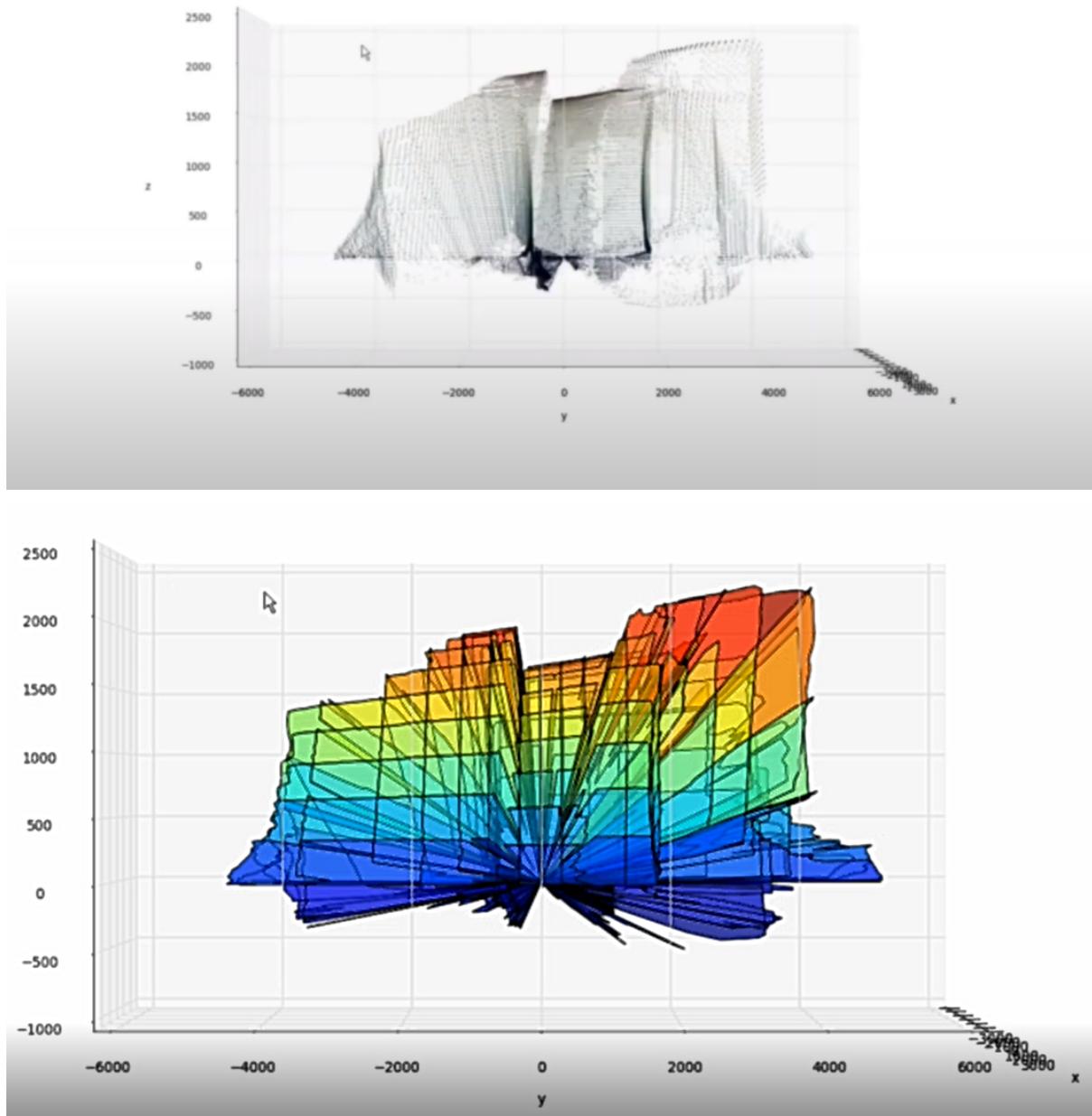
Scan & Normalize Coordinates

```
122 #Prerequisites for 3D Simulation
123 theta = np.linspace(0,2*np.pi,360,dtype='float16')
124 phi = np.linspace(0,np.pi,180,dtype='float16')
125 xx = np.outer(np.cos(theta),np.sin(phi))
126 yy = np.outer(np.sin(theta),np.sin(phi))
127 zz = np.outer(np.ones(np.size(theta)),np.cos(phi))
128
129 #scan with respect to per degree
130 for i in range(180):
131     print("degree: ", str(i))
132     board.digital[pin].write(i)
133     sleep(0.1)
134     board.digital[pin].write(i)
135     sleep(1)
136     data1 = next(gen)
137     data2 = next(gen)
138     data3 = next(gen)
139     data = mean(data1, data2, data3)
140     plot(i, data)
141
90 #coordinates normalization
91 def plot(i, data):
92     tar1 = i
93     tar2 = i + 180
94     for j in range(0,180):
95         xx[tar1,j] = xx[tar1,j]*data[j]
96         yy[tar1,j] = yy[tar1,j]*data[j]
97         zz[tar1,j] = zz[tar1,j]*data[j]
98         xx[tar2,j] = xx[tar2,j]*data[359-j]
99         yy[tar2,j] = yy[tar2,j]*data[359-j]
100        zz[tar2,j] = zz[tar2,j]*data[359-j]
```

3D Simulation

```
55 for i in range(0,390,30):
56     ax.cla()
57     ax.set_xlabel('x')
58     ax.set_ylabel('y')
59     ax.set_zlabel('z')
60     ax.view_init(0,i)
61     ax.plot_surface(x[0],z[0],y[0],cmap='copper',alpha=0.5)
62     plt.pause(0.5)
63 plt.close()
```

Implementation Result:



上面是以scatter3D模擬出的空間，下圖是以surface模擬出的空間，考量到視覺的方便性，最我們偏向選擇以scatter3D做我們的最終輸出。

Demo Video:

<https://www.youtube.com/watch?v=djTdtZT3nL4>

Introduction Video:

<https://www.youtube.com/watch?v=A1vDghdHM0Y>

Work Division:

陳泳翰 - 光達輸出轉置成矩陣、client to host聯絡架設

吳峻陞 - 硬體 & 開發板環境架設、光達掃描矩陣數值優化

共同 - 3D空間模擬演算法

Reference:

<https://reurl.cc/Nrqybm> - VNC Viewer在Jetson Nano中設定說明

<https://reurl.cc/rgL6E1> - iptables使用 (因為Jetson Nano不在學校, 需要使用外網連接使用VNC Viewer, 其中要設定防火牆才能成功開啟遠端連線)

<https://www.youtube.com/watch?v=8j3Fo-16Rr8&t=348s> - PyFirmata相關使用和連接

<https://github.com/lakshmanmallidi/PyLidar3> - PyLidar3 library

<https://clay-atlas.com/blog/2019/10/15/python-chinese-tutorial-socket-tcp-ip/> - socket使用

<https://officeguide.cc/python-matplotlib-three-dimensional-plotting-tutorial-examples/> -3D plot