December 14, 2021

LUND UNIVERSITY

# A Holodeck for Dung Beetles

By

Xi Chen
xi7224ch-s@student.lu.se
Xingda Li
xi6341li-s@student.lu.se
Haidi Hu
ha3077hu-s@student.lu.se
Sijia Cheng
si4168ch-s@student.lu.se

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund Univeristy
SE-221 00 Lund, Sweden

2021

Project in Computer Science - EDAN70
Supervisor(s): Flavius Gruian, Department of Computer Science
Yakir Gagnon, Department of Biology

# Contents

# 1 User Manual

The user manual describes how to use our holodeck system for redirecting dung beetles. And the corresponding github link is: https://github.com/eziohhd/EDAN70_CS_PROJECT/blob/main/User%20Manual.md

## 1.1 Hardware setup

The hardware setup has already been done. In case of accidents, web pages related to hardware settings are placed here.

- LIDAR: https://buy.garmin.com/en-US/US/p/578152
- Turret, servos, U2D2, microcontroller, and more: https://www.trossenrobotics.com/p/ScorpionX-RX-64-robot-turret.aspx
- Laser: https://se.rs-online.com/web/p/laser-modules/1271557

## 1.2 Software setup

### 1.2.1 Arduino

For Arduino, the executable file has been written and does not need to be changed. However, if you want to make some modifications to the executable file, you can modify the DungBeetle.ino file in the DungBeetle folder. If you need to rewrite the ino file into the arduino, you can check this web page to find what you might need :https://learn.trossenrobotics.com/arbotix/7-arbotix-quick-start-guide

### 1.2.2 Raspberry Pi

In our system, we use python scripts to control the Raspberry pi. And the environment and libraries required for the program to run have been configured. In case of Raspberry pi is damaged, you can follow the instructions below to install the required libraries.

- OpenCV: https://qengineering.eu/install-opencv-4.5-on-raspberry-pi-4.html
- Inquirer:

```
pip3 install inquirer
```

- Pandas:

```
sudo apt-get install python3-pandas
```

- Numpy:

```
sudo apt-get install python3-numpy
```

## 1.3 Instructions about how to run the system

- After connecting the USB port of the arduino to the Raspberry Pi, power on the two microcontrollers.
- Use ssh to control the Raspberry pi remotely
- When you connect to the raspberry pi, run this command to enable the pi camera(You only need to do it once after the Raspberry Pi is powered on):

```
libcamera-jpeg -o test.jpg
```

- Find the .py file and execute:

```
python3 Desktop/Raspberry/calibration_test.py
```

- Select the mode you want:

  When you place the system in a new room, the first thing you need to do is scanning. After the scanning, you need to do the calibration. The last step is Beetle_redirection. To save your time, you can bypass the scanning and choose calibration if the camera is moved but the system is still in the same place. If the location of the system is exactly where the system was initialized, you can bypass scanning and calibration and choose Beetle_redirection.

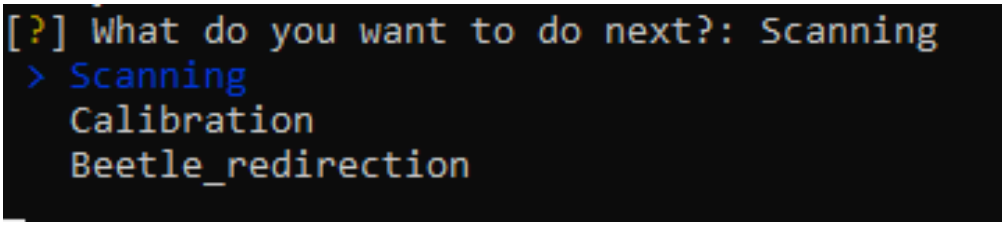


**Figure 1** – mode selection

- Scanning:

  In this mode, the Lidar will scan the whole room and send the pan&tilt angle data together with the distance data to the Raspberry Pi via serial communication. When the scanning is done, a *scandone* signal will be printed in the terminal which means the scanning process is completed. The data obtained by scanning is stored in $points\_cloud.csv$ which can be visualized by using *construction.m* in *matlab* folder.

- Calibration:

  In this mode, the laser will randomly point to a certain area on the roof(the area can be modified by changing the index range) to do the calibration. Moreover, you can define the number of points you would like to take in calibration by setting the value of variable $number\_of\_samples$ in the code. When the calibration is done, the system will automatically enter the test mode to verify the performance of calibration by comparing the difference between the calculated location of the laser point and the actual location of the laser point. During the calibration, the transformation matrix will be written into $Calibration\_matrix.csv$ for further use.

- Beetle_redirection:

  Right now this part is just a demo for test as we don't have a real dung beetle to complete our algorithm. However, with the help of pi camera, you can easily get the location of the dung beetle in the image and redirect it by using the algorithm in this mode.

## 2 Collaboration

This section specify everyone's contribution to the work.

**Xi Chen**

- Mapping algorithm and implementation
- Evaluation of the calibration model

**Haidi Hu**

- Redirecting the dung beetle with laser
- Evaluation of the redirecting model

**Xingda Li**

- Communication between Pi and $\mu$C
- Mapping algorithm and implementation

**Sijia Cheng**

- Controlling the laser with Pi
- Construction of the 3D model of the room

# 3 Introduction

Dung beetles navigate using the sun by maintaining a constant azimuth (relative to the sun) while walking. And the aim of the project is to simulate the sun with a bright laser for dung beetles in an enclosed room and control their walking direction. The project can be roughly divided into three tasks: construction of the 3D model of the room, mapping between 2D camera and 3D LIDAR and Redirecting the dung beetle with laser. The schematic of the system is shown in Fig.2.

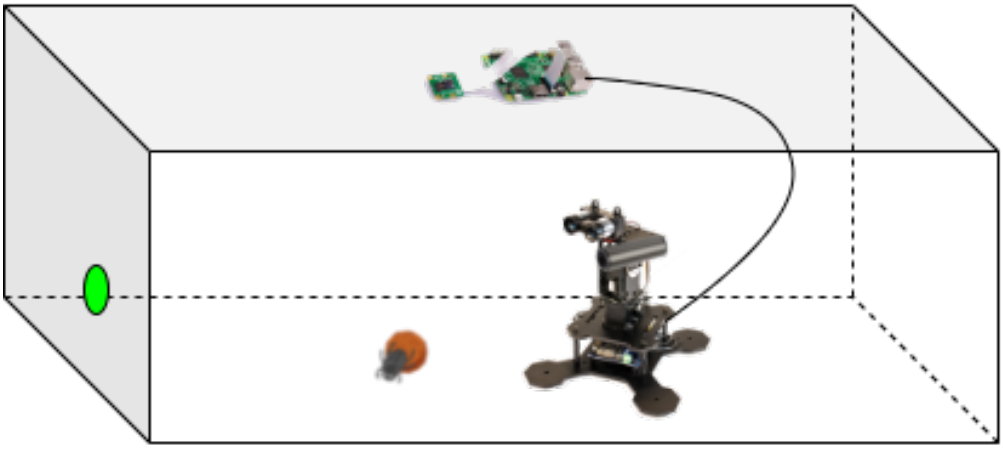

**Figure 2** – Schematic of the setup in real scenario

The first task, construction of the 3D model of the room, is to obtain the geometry of the room. Since the the system is going to be deployed in Sweden, Sardinia, and South Africa, it needs to be flexible enough to work in a variety of environments, which means that it has to adapt to rooms with different geometries. To this end, as shown in Fig.2, a LIDAR sensor with a pan and tilt turret(powered by two servo motors) will help us with getting the 3D model of the room.

The second task, mapping between 2D camera and 3D LIDAR, is to calibrate the camera and the laser-LIDAR turret. To identify the location of the beetles in the room, the project requires the camera and laser-LIDAR turret to communicate and calibrate each other. Like in Fig.2, the Pi camera with the Raspberry Pi is mounted on the ceiling and connected to the Arbotix-M micro controller. However, in this project, the Pi camera will be mounted on

the floor due to the limited length of the cable.

The last task, redirecting the dung beetle with laser, is to change or maintain the walking direction of the beetles. The laser's point will reflect off of the room's walls and function as an artificial sun for the beetle. By dynamically moving the laser pointer, we can redirect the walking direction of the beetle in real time. Using the trajectory of the beetle on the floor in combination with the known geometry of the room, the azimuth of the apparent sun can be calculated.

# 4 Implementation

## 4.1 Construction of 3D Room Model Using LIDAR

The geometry of the enclosed room is obtained using a LIDAR sensor with a turret. The LIDAR sensor can tell the distance while the turret gives us the pan and tilt angle of the LIDAR sensor, which is sufficient for getting a coordinate on the outline of the room. Thus, by scanning the whole room, we are able to construct the 3D model of the room.

As shown in the Listing 1, the pan of Lidar swings between 0° and 360° back and forth. When the pan of the Lidar reaches either 0°(PAN_MIN) or 360°(PAN_MAX), the angle of elevation increases by a certain value.

```
if(servoMoving1 == 0)//Pan stop moving when it reaches PAN_MIN or PAN_MAX
  {
    servoMoving2 = ax12GetRegister(2, AX_MOVING, 1);
    tiltCurrentPosition = ax12GetRegister(2, AX_PRESENT_POSITION_L, 2);
    if(tiltCurrentPosition >= TILT_MIN && tiltCurrentPosition <= TILT_MAX)
    {
    tiltGoalPositon = tiltCurrentPosition + 20;
    if(panGoalPositon == PAN_MIN)
    {
      panGoalPositon = PAN_MAX;
    }
    else
    {
      panGoalPositon = PAN_MIN;
    }
    delay(5);
    SetPosition(1,panGoalPositon);
    SetPosition(2,tiltGoalPositon);
    servoMoving1 = 1;
  }
```

**Listing 1** – Code for scanning the room in C++

Meanwhile, the Arbotix-M Micro Controller receives the measurements from LIDAR sensor and the pan&tile turret and delivers them to Raspberry Pi via serial communication.

After the scanning completes, point clouds that depict the 3D model of the room are obtained and stored in Raspberry Pi for further uses. The 3D plot of these point clouds would look similar to Figure.3

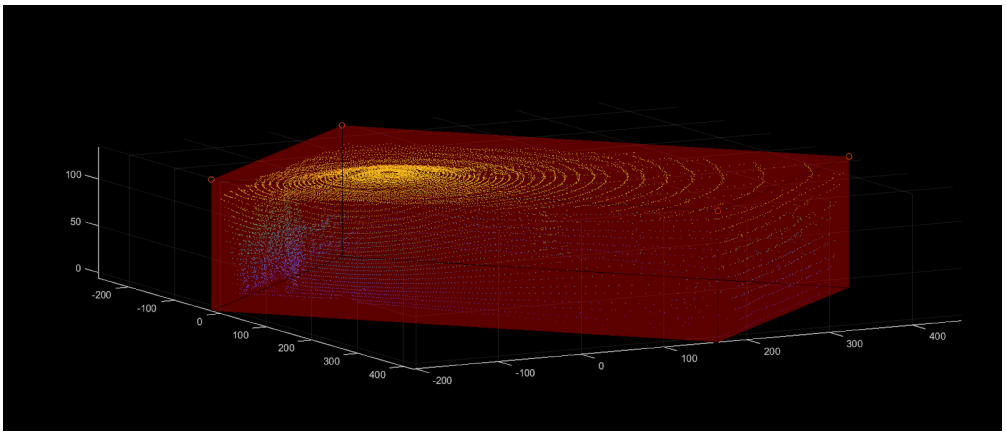


**Figure 3** – 3D plot of the point clouds from room scanning

## 4.2 Calibration Model for 2D camera and 3D LIDAR

As mentioned above, there is a Pi camera that is mounted on the floor and facing the ceiling of the room. After getting the point clouds, it's time to calibrate the 2D camera and the 3D LIDAR. The image data captured by the camera are formed by two-dimensional coordinate system(U,V) and the range data of the 3D point clouds are represented by three-dimensional coordinate system(X,Y,Z). The transformation from a 3D point(x,y,z) to a 2D point(u,v) can be represented by:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} = M \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{1}$$

where $f_u$ and $f_v$ are the effective focal lengths in horizontal and vertical directions, respectively, and ($u_0$, $v_0$) is the center point of the image plane. R and t are the rotation and the translation matrices[1].

As there is a linear relation between these two coordinate system, the transformation function eq.(1) can be modified as:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = M \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \\ m_{41} & m_{42} & m_{43} \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{2}$$

Note that the transformation Matrix M in eq.(2) is not the same as that in eq.(1).

Eq.(2) can be rewritten as the following equations:

$$x = \frac{m_{11}u + m_{12}v + m_{13}}{m_{41}u + m_{42}v + m_{43}} \tag{3}$$

$$y = \frac{m_{21}u + m_{22}v + m_{23}}{m_{41}u + m_{42}v + m_{43}} \tag{4}$$

$$z = \frac{m_{31}u + m_{32}v + m_{33}}{m_{41}u + m_{42}v + m_{43}} \tag{5}$$

and in the form of matrix multiplication as:

$$\begin{pmatrix} u & v & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -ux & -vx & -x \\ 0 & 0 & 0 & u & v & 1 & 0 & 0 & 0 & -uy & -vy & -y \\ 0 & 0 & 0 & 0 & 0 & 0 & u & v & 1 & -uz & -vz & -z \end{pmatrix} \begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{6}$$

As in eq.(6), for each corresponding pair there are three equations. Therefore, to solve M, we need at least 4 corresponding pairs. However, in the implementation, we take 30 pairs in order to get a more accurate transformation matrix. To solve this homogeneous linear equation system, we apply the singular value decomposition(SVD) method.

The first step of the calibration is to take a certain number points from point clouds we got from the first task. Next, the pans and tilts corresponding to these points are feed to the servo motors one by one. Once the servos are in desired position, the camera detect the 2D position of laser point in the image. Then, this 2D point and the 3D point from point clouds which is the correspondence pair will be stored. This process will iterate until the certain number of correspondence pairs are obtained.

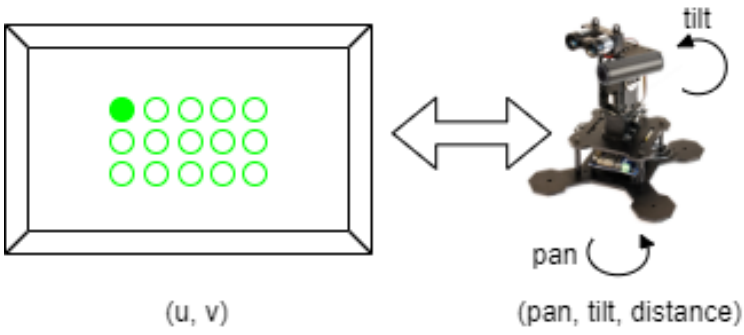


**Figure 4** – Calibration configuration of image taken by 2D camera and 3D LIDAR

The process of getting corresponding pairs is depicted in Figure 4. The left of Figure 4 shows the image taken by the camera and the solid green circle on it represents the laser point while the hollow ones represent the positions of the laser point in the following iterations respectively.

The Python code for storing correspondence pairs in the form of the multiplicand in eq.(6) is shown in Listing 2 and that for solving the homogeneous linear equation system with the SVD method is shown in Listing 3.

```
# Store corresponding pairs
def Matrix(x,y,z,u,v,A):
    coe1 = [u, v, 1, 0, 0, 0, 0, 0, 0, -u*x, -v*x, -x]
    coe2 = [0, 0, 0, u, v, 1, 0, 0, 0, -u*y, -v*y, -y]
    coe3 = [0, 0, 0, 0, 0, 0, u, v, 1, -u*z, -v*z, -z]
    A.append(coe1)
    A.append(coe2)
    A.append(coe3)
    return A
```

**Listing 2** – Code for storing correspondence pairs in Python

```
# Solve a Homogeneous Linear Equation System: Ax=0
def sol_svd(A):
    A = np.array(A)
    # find the eigenvalues and eigenvector of U(transpose).U
    e_vals, e_vecs = np.linalg.eig(np.dot(A.T, A))
    # extract the eigenvector (column) associated with the minimum
   eigenvalue
    return e_vecs[:, np.argmin(e_vals)]
```

**Listing 3** – Code for SVD in Python

With solving the linear system, the transformation matrix M now is available and the transformation from a 2D point(x,y,z) to a 3D point(u,v) can be done using eq.(3), eq.(4) and eq.(5). The Python code that realizes this transformation is shown in listing 4

```
# get corresponding coordinate after getting M
def get_corr_point(M,u,v):
    M = M.reshape((4,3))
    x = (M[0][0]*u + M[0][1]*v + M[0][2])/(M[3][0]*u + M[3][1]*v + M
   [3][2])
    y = (M[1][0]*u + M[1][1]*v + M[1][2])/(M[3][0]*u + M[3][1]*v + M
   [3][2])
    z = (M[2][0]*u + M[2][1]*v + M[2][2])/(M[3][0]*u + M[3][1]*v + M
   [3][2])
    p3 = np.array([[x],[y],[z]])
    return p3
```

**Listing 4** – Code for transformation from a 2D point to a 3D point in Python

## 4.3 Redirecting Dung Beetle with the System

In order to control the trajectory of the dung beetle, the whole process could be divided into four steps. Firstly, the location of the dung beetle in the image can be got by using pi camera. Secondly, the coordinates of the dung beetle in the image could be converted into coordinates in the 3D points cloud by using the transfer matrix. Thirdly, when the dung beetle moves, we take the current coordinate of the dung beetle as the origin and the dung beetle's movement direction as the positive X-axis to recalculate the position of all the points in the 3D points cloud. The last step is to enter the relative angle relationship between the laser point and the dung beetle to get the laser target position and the corresponding pan&tilt data.

```
#Use random numbers to represent the location of a real dung beetle
random_x = random.randrange(0,640)
random_y = random.randrange(0,480)
#Transfer the location in the image to the 3d cloud points
random_location = get_corr_point(M, random_x, random_y)
beetle_location_x = random_location[0]
beetle_location_y = random_location[1]
beetle_location = [beetle_location_x,beetle_location_y]
#Recalculate the position of all the points in the 3D points cloud
new_points = ConvertXYZ(beetle_location,points);
#Take out the points we want and send the pan&tilt data to the motor
angle_distance = GetAngle(new_points)
[index,laser_target] = GetLaserTarget(direction,angle_distance,points)
pan,tilt = GetPanTilt(index,rawM)
MotorControl(pan,tilt)
```

**Listing 5** – Code for dung beetle redirection in python

# 5 Evaluation

## 5.1 Evaluate the Accuracy of the Calibration

After obtaining the transformation matrix M in calibration, we would like to evaluate the accuracy of the calibration. There are two groups in this evaluation, one with the last 300 points(smaller area) from point clouds and the other with the last 1000 points(bigger area). The reasons why we take the last a few points is that we are only interested in the points on the roof of the room model and also, the area seen by the camera is limited. Plots of the last 300 and 1000 points from point clouds are shown in Figure 5 and Figure 6, respectively.

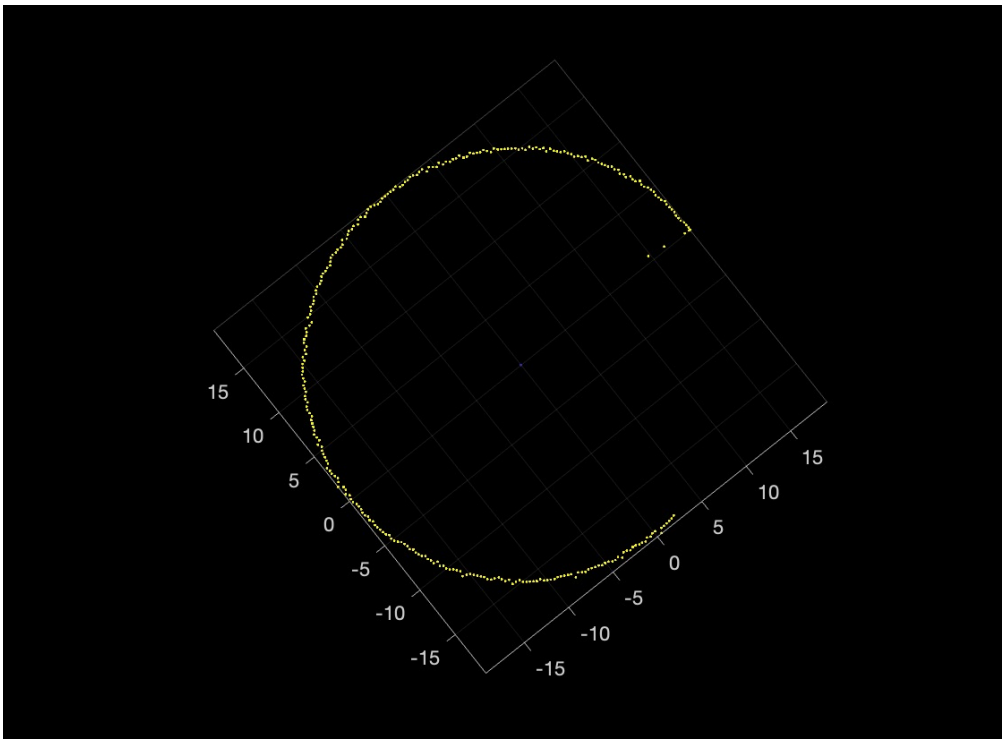


**Figure 5** – Plot of the last 300 points from point clouds

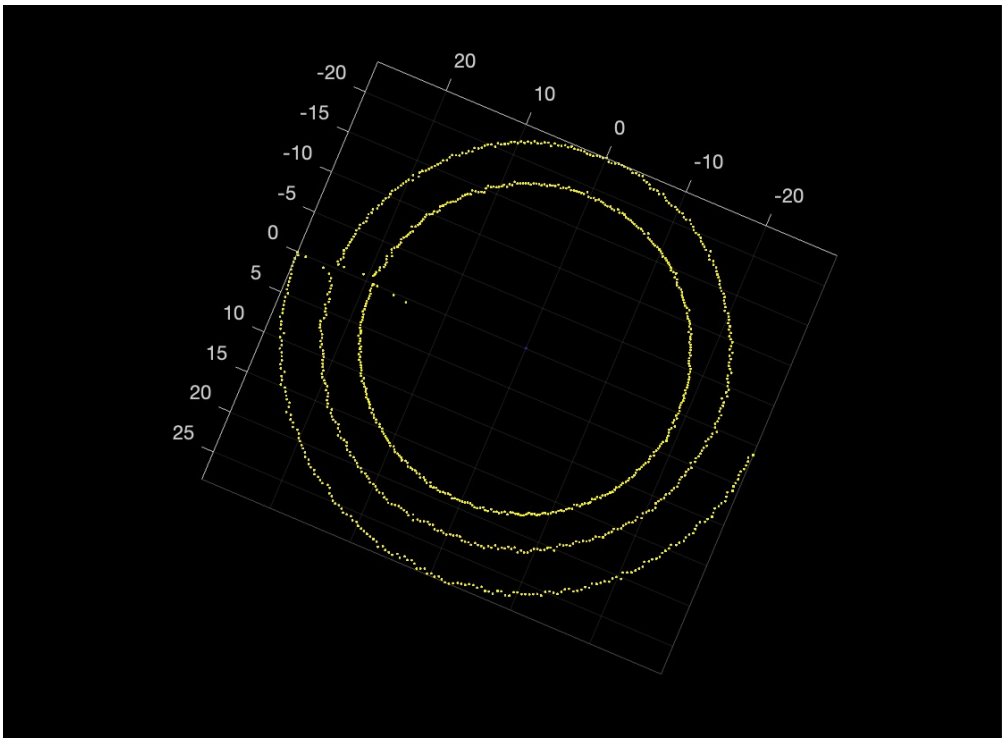


**Figure 6** – Plot of the last 1000 points from point clouds

Figure 7 illustrates how we evaluate the calibration model. First of all, we randomly pick one point from either the 300 points or 1000 points as we mentioned before. Then, by feeding the pan&tilt corresponding to that point to the turret with the laser on, the servos will move to the position with a laser point on the roof. With the Pi camera facing the roof, the 2D position of the laser point is detected and transformed to the 3D position in the room model using the transformation matrix we obtained in calibration. Finally, the error of the calibration model is obtained by calculating the distance between these two 3D points. In Figure 7, the red dot represents the point from point clouds while the green one represents the laser point.

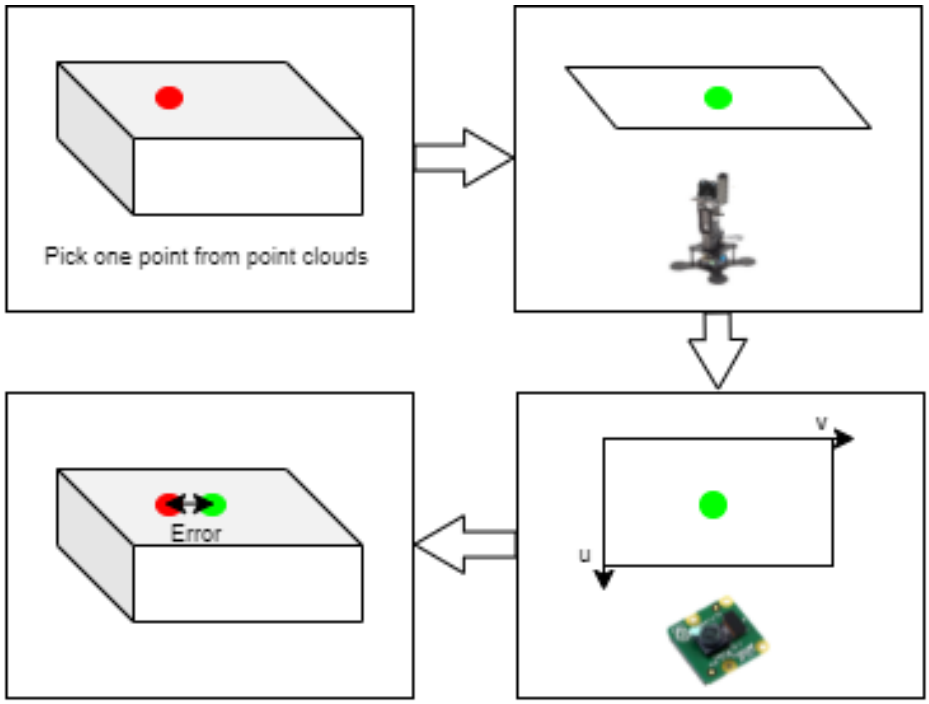


**Figure 7** – Illustration of evaluating the calibration model

The evaluation results is given in Table 1. There are six time steps as we iterated the process above for six times. And the "smaller area" and "bigger area" indicates either it's the group using 300 points or the one using 1000 points. For example, in the "bigger area" group, we took the last 1000 points from point clouds as shown in Figure 6 and only the points within these 1000 points were used to calibrate and evaluate the calibration model.

**Table 1** – Evaluation results for calibration

| Time step | Error(cm) Smaller area | Error(cm) Bigger area |
|---|---|---|
| 1 | 1.8198 | 6.0012 |
| 2 | 1.6297 | 2.8394 |
| 3 | 1.2211 | 2.2814 |
| 4 | 0.8160 | 1.5990 |
| 5 | 0.8710 | 1.3283 |
| 6 | 1.3383 | 1.9129 |

## 5.2 Test the System with a Random Pattern

Here we use random numbers to get the position of the dung beetle, and move the laser point to a certain point in the room that is in a fixed angle relationship with the dung beetle, and

calculate the time required for the motor to move.

**Table 2** – Evaluation for random pattern movement

| Time step | Starting point(cm) | Ending point(cm) | Time(ms) |
|---|---|---|---|
| 1 | (47,-129,167) | (125,573,42) | 9442 |
| 2 | (125,573,42) | (22,32,169) | 3298 |
| 3 | (22,32,169) | (-13,20,172) | 3091 |
| 4 | (-13,20,172) | (-91,-28,173) | 3304 |

## 5.3 Test the System with a Constant Pattern

Here we let the dung beetle move along y=x, and each time x and y are increased by 10cm. In order to facilitate observation, the angle relationship between the laser point and the dung beetle is set to [0, pi/2], so the position of the laser point is always right above the dung beetle.

**Table 3** – Evaluation for constant pattern movement

| Time step | Starting point(cm) | Ending point(cm) | Time(ms) |
|---|---|---|---|
| 1 | (10,16,172) | (20,27,169) | 327 |
| 2 | (20,27,169) | (30,33,170) | 345 |
| 3 | (30,33,170) | (40,46,169) | 317 |
| 4 | (40,46,169) | (50,63,172) | 326 |

# 6 Conclusion and Discussion

In evaluation, the trajectory of the beetle can be detected and the azimuth of the simulated sun can be calculated in real time. The accuracy of the calibration model looks reasonable, but with the increase of the area that taken into the consideration, the error tends to increase. As for the entire system, it's difficult to test as we don't have the dung beetle. Therefore, the accuracy of the system need to be further investigated and some further work need to be carried out.

In theory, the more correspondence pairs fed to the calibration model, the more accurate the calibration should be. Thus, by calibrating with all the points that on the roof of the room model, the error can be minimized. However, this process might take a long time. Therefore, investigation to find the most suitable number of correspondence pairs might take place.

As we mentioned before, the area seen by the camera is limited due to the length to height ratio of the room. An extra lens might be applied to the camera and non-linearity might be introduced. Thus, we might need to update the transformation solution during calibration by using a nonlinear least squares method to tackle the non-linearity issue.

Furthermore, the algorithm to determine the desired direction of the dung beetle is needed. In our work, we just evaluated the system with some patterns that indicate the parameters of dung beetle. To complete the system, this algorithm need to be implemented.

# References

[1] Park, Yoonsu & Yun, Seokmin & Won, Chee & Cho, Kyungeun & Um, Kyhyun & Sim, Sungdae. (2014). Calibration between Color Camera and 3D LIDAR Instruments with a Polygonal Planar Board. Sensors (Basel, Switzerland). 14. 5333-5353. 10.3390/s140305333.