UAV ground detection and tracking systems: Tree detection and tracking change species and amount of tree

Christian Ekeigwe
Purdue University
Computer Information Technology, Cybersecurity
West Lafayette, Indiana, United States
cekeigw@purdue.edu

Daehyeon Jeong
Pusan National University
Aerospace Engineering
Pusan, Republic of Korea
empcik@pusan.ac.kr

Jaeyoung Shim
Pusan National University
Computer Engineering
Pusan, Republic of Korea
jysim0129@pusan.ac.kr

Jeonghwan Kang
Pusan National University
Computer Engineering
Pusan, Republic of Korea
chanelacy@outlook.com

Seoungheong Jeong
Pusan National University
Computer Engineering
Pusan, Republic of Korea
korssouna1030@pusan.ac.kr

***Abstract*—This paper examines the technic and algorithm about detect tree using Unmanned Aerial Vehicles (UAV) in real-time. The purpose is grasp distribution of tree species using real-time air-view videos contain GPS information. This examination use 'Haar Cascades'[1] to detect tree and using Raspberry Pi to receive real-time video data.**

This examination shows 89% of accuracy in when using 17000 units of data.

The limitation of examination is 'Haar Cascades' need enough data to detect tree, but this paper examines only 1700 units of data, so the accuracy is little low. And Raspberry Pi CPU core speed is slow so that can't handle high resolution video in real-time.

Keywords—OpenCV, Haar Cascades, UAV, tree detection

I. Introduction

The BGCI(Botanic Gardens Conservation International) estimated that at least 11,000 tree species are threatened with extinction in the wild[2]. This means the importance of tracking the number of tree species that change has increased.

The accurate characterisation of tree species distribution in forest areas is a significant task for forest management and forest research. In particular, management and protection of native vegetation[3], monitoring invasive species[4], mapping wildlife habitats[5], and sustainable forest management[6] are some of the goals of research that require tree species distribution characterisation[7] on a wide scale. To this end, many studies have been conducted using remote sensing data.[8]

The use of remote sensed data in forest management is a common practice since it allows the collection of a large amount of information about the environment with less human effort in the field. This is even more important in large or inaccessible areas, where the fieldwork is more exhausting and expensive.

In this examination design a novel way to fly over a tree by UAV with a camera on it and determine what kind of tree it is. Then record the GPS point of the tree. At the endpoint of a flight, a map will show all of the GPS points of the tree.

Detect tree amount and classify tree species with OpenCV 'Haar Cascades'. 'Haar Cascades' needs a lot of positive images (images of trees) and negative images (images without trees) to train the classifier. This examination uses 7 units of video. Each of the videos is split into 253 units of images. And convert color-image to gray-image for clearness of contour line. This attempt to detect tree using 'Haar Cascades' shows approximately 89% of accuracy.

The examination uses Raspberry Pi and camera module instead of using UAV to test real-time processing of classifying tree data indirectly. The advantage of raspberry pie is that it is small and light and allows remote access over the network. Connect to Raspberry Pie remotely and get real-time air-view video of tree. The data values obtained through the process are stored in a database such as MariaDB or MySQL.[9] The goal of this project is to bring up records stored in that database and visualize them. There will be many ways to visualize the data, but intuitively, this project considering marking the types of trees by color on the map. If the data sort out trees and obtain the data of the latitude and longitude for each tree, it is available to mark on a map with Python 'folium' library. As a result, classified data show all of the GPS points for a specific variety of trees.

This technic makes the possibility to contribute detect trees and find new species or tracking changes of tree environment for UAV consumer, not commercial purpose user. In addition, government or Environment groups can use this technic for tracking tree species and amounts easily. But because of small units of data examination shows low tree detect accuracy.

II. Materials and Methods

A. Materials

The subject of this examination is distinguish tree species and detect tree and its amount using artificial intelligence. This examination use weak artificial intelligence(AI). The weak AI is artificial that is limited to a specific or narrow area. And that's the reason why this examination use weak AI to detect and classify trees.

B. Methods

For detect trees and its species, in this examination use Haar feature-based cascade classifiers. Many positive and negative images needed for detect stuff well. This examination use 00 units of positive image and 00 units of negative images. The positive image is part of UAV video that sliced with 0.05 second. This images convert to gray-image for clearness of contour line. And for the best result, this examination use OpenCV Cascade Classifier which is well-known for easy to use. To see the real-time detect of tree, Raspberry Pie with camera module is prepared. And access to the Raspberry Pie and receive air-view video data that contains GPS data. At the end classify tree amount and species.

III. Results and Disccustion

This examination divides 3 step. First one is finding the way to how to detect tree. Second step is gathering data for training and increase accuracy. The final step is conduct real-time tree detection using UAV.

A. *Method for tree detection*

This paper examines two methods for detect tree. First one is edge detection. Edge detection process serves to simplify the analysis of images by drastically reducing the amount of data to be processed, while at the same time preserving useful structural information about object boundaries [10]. The first step for Canny edge detection is eliminate noise because edge detection results are highly sensitive to image noise. Gaussian blurring is used for decreasing noise. This method can find best line for find tree edge by adjust sigma value and Gaussian kernel. And finally use Non-Maximum Suppression for detect tree but the parameter value is wide video to video, so this method (Canny edge detection) is not proper for detect tree. As a result, this method is useful to distinguish tree or not but there are two problems. One is that it is hard to make criterion for detect tree. The second problem is handling with each tree leaf shape because the tree leaf shape is diverse tree by tree.

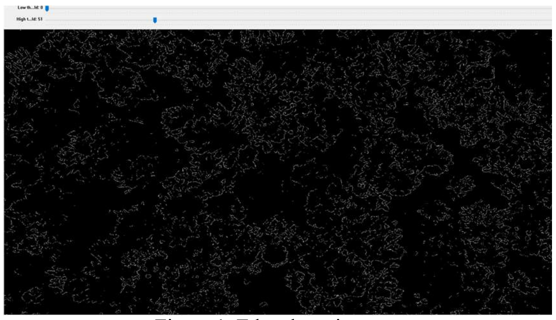


Figure 1. Edge detection

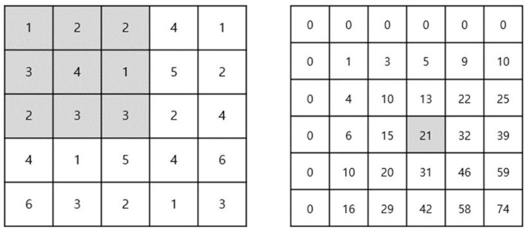
So, in this paper examines processed with second solution, 'Haar Cascades.' The process of 'Haar Cascades' is consist with 4 steps. The first step is selecting Haar Features. If use 24x24 size image, over 160000 amounts of Haar Features are calculated. The Haar Feature has three characteristics.

1) *Edge Features:* The value of Haar Feature consist with two sqaure is calculated by subtraction with the pixels that black side ans white side.

2) *Line Features:* The value of Haar Feature consist with three sqaure is calculated by subtraction with the pixels that one inside sqaure black side and two outside sqaure outside.

3) *Four-rectangle Features:* The value of Haar Feature consist with four sqaure is difference with the pixels that diagonal side of.

'Haar Cascades' is extremely rapidly and achieving high detection rates. The reason why the 'Haar Cascades' fast is Integral Images. To make Integral Images, at first, add one to original image's width and height. And then set area of original image and get combination value. And input the value to diagonal pixel is respond.

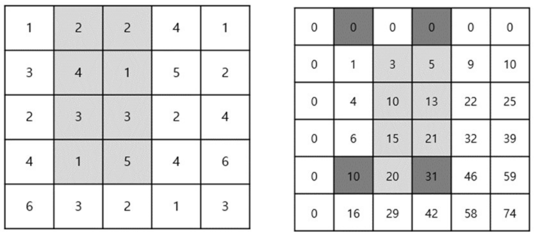


1	2	2	4	1
3	4	1	5	2
2	3	3	2	4
4	1	5	4	6
6	3	2	1	3

0	0	0	0	0	0
0	1	3	5	9	10
0	4	10	13	22	25
0	6	15	21	32	39
0	10	20	31	46	59
0	16	29	42	58	74

Figure 2. Integral Images 1

To calculate form left side of image (the original image) Integral Images can calculate with 4 respond area pixel value. In Integral Images correspond area pixel, subtract right bottom pixel with right and top side pixel and add diagonal side pixel.



1	2	2	4	1
3	4	1	5	2
2	3	3	2	4
4	1	5	4	6
6	3	2	1	3

0	0	0	0	0	0
0	1	3	5	9	10
0	4	10	13	22	25
0	6	15	21	32	39
0	10	20	31	46	59
0	16	29	42	58	74

Figure 3. Integral Images 2

Adaboost is useful method to power up the performance. Most feature that calculated in Integral Images are useless information. For example, the meaningful information in tree, is edge side's difference of bright (Figure 4).

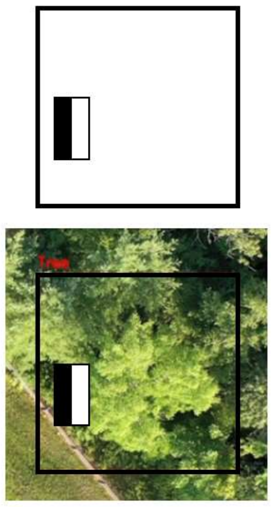


Figure 4. Tree detection example

The Adaboost process repeat 3 steps.

1) *Setting equal threshold:* At first, every Haar Features have same threshold.

2) *Find low error:* Using the result that trained with every Haar Feature data set, calculate each Haar Features' error rate and incorrectly classified Haar Features increase the weight.

As a result, better performance Haar Features have low error rate.

3) *Repeat step 2:* To find demended feature or error rate, repeat step 2.

In 'Haar Cascades' process, if find low error rate Haar features then use Cascade Classifier. Most of area of images are empty-tree space. So current window try to detect tree, and judge that is not a tree, then the window move to next area. But if the current window judge that is a tree, then comparison with other Haar Features. This is Haar Classifier and it can make fast decision for detect images.

The key insight of cascade of classifier is that smaller, and therefore more efficient, boosted classifier can be constructed which reject many of the negative sub-windows while detecting almost all positive instances. It means that for tree detection, it only needs enough positive and negative image.

B. Gathering data and increase accuracy

At first, this examination tries to use Google Map because it has many data set but Google Map has only low resolution so the data set should be gathering in other way.

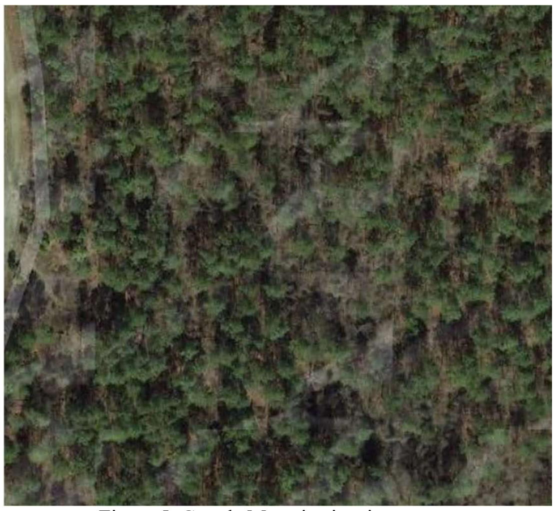


Figure 5. Google Map air-view image

Using UAV take 7 units of video of tree air-view contains 107 units of positive and 146 units of negative data.

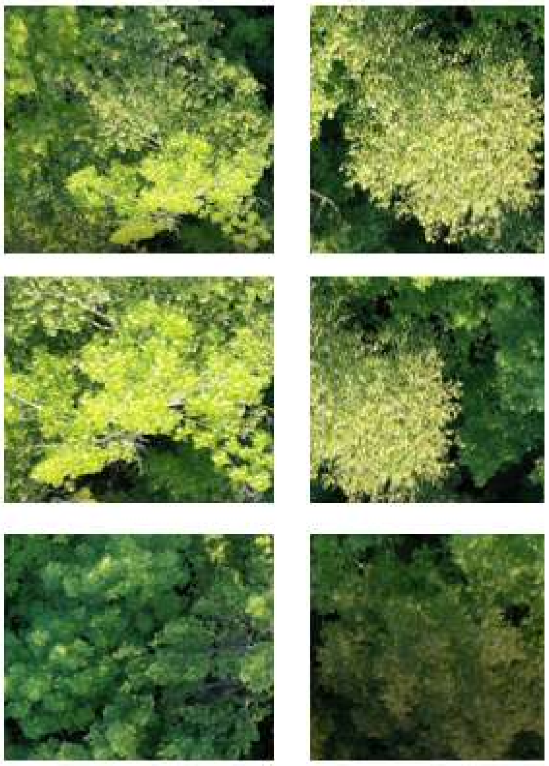


Figure 6. Positive data from UAV video

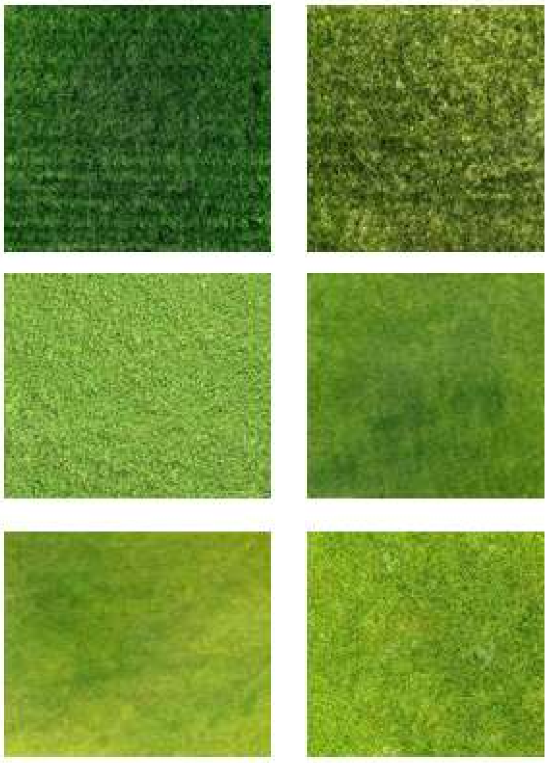
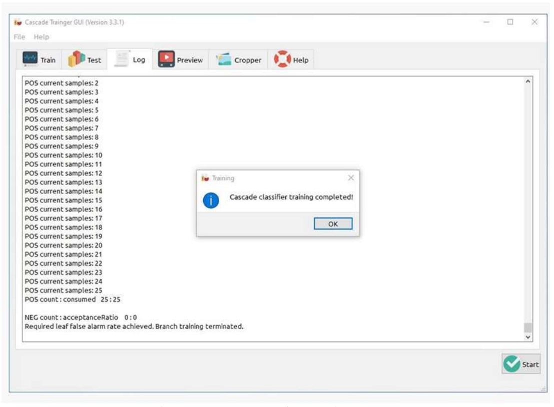


Figure 7. Negative data from UAV video

This examination use 'Cascade Trainer' version 3.3.1 for easy training. It helps selecting training data and has automated training system.



```
POS current samples: 2
POS current samples: 3
POS current samples: 4
POS current samples: 5
POS current samples: 6
POS current samples: 7
POS current samples: 8
POS current samples: 9
POS current samples: 10
POS current samples: 11
POS current samples: 12
POS current samples: 13
POS current samples: 14
POS current samples: 15
POS current samples: 16
POS current samples: 17
POS current samples: 18
POS current samples: 19
POS current samples: 20
POS current samples: 21
POS current samples: 22
POS current samples: 23
POS current samples: 24
POS current samples: 25
POS count : consumed   25 : 25

NEG count : acceptanceRatio   0 : 0
Required leaf false alarm rate achieved. Branch training terminated.
```

Cascade classifier training completed!

Figure 8. Cascade Trainer

The negative images required by the 'Cascade Trainer' are about 1,000 and the positive images require an appropriate amount of sample data. Therefore, a separate program was developed to secure a large amount of data. The program provides the following functions. It captures and saves the image in the area designed by the user while playing the video. User can specify the image in the red box as shown in the figure. And user can save the image by setting image in the designated area to positive or negative.

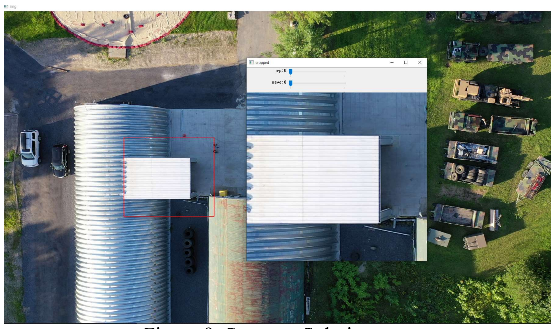


Figure 9. Separate Solution

The saved images area divided and saved in p and n folders. The positive image saved with program was 107 samples and the negative 146 samples were used for learning. Although it is less than the 1000 negative images required by the program. Considering the amount of data this examination has, it is an appropriate amount.

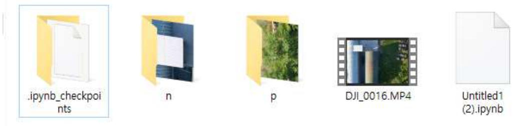


Figure 10. Separate Solution

At the end of training, check the result of training is necessary. Writing code for conduct the result of training. This examination uses the OpenCV with Python. The code means capture video, slice it with 0.05 second, convert to gray-image, detect tree using trained data, and show by rectangle image cover tree.

C. *Real-time tree detection using UAV*

Real-time tree detection using UAV demonstration conducted with Raspberry Pi. The main idea is detecting from Raspberry Pi, store to SQL and the extraction and visualization.

Detecting from Raspberry is conducted with detecting model that trained with 'Haar cascade.' To find best result, this paper examines with four resolution videos.

Resolution	3840x2160	1920x1080	1280x720
Bit-rate	102132	160251	76133
Frame	29.97	29.97	29.97
Calculation time	478 second	133 second	71 second

Figure 11. Cascade Trainer

Unusually HD(1280x720) resolution has best and most accuracy. This progress can find amount of tree at specific latitude and longitude position.

1) Raspberry Pi4 Environment Setting: Libraries including openCV 4.5.3, MariaDB 10.3.29, mysql 1.0.5 (SQL management), xrdp 0.9.9-1 (Remote control management), gpsd 3.17-7 (Connecting GPS module).

2) Raspberry Pi4 Equipped modules: RPI 8MP CAMERA BOARD V2(Camera module), NEO-7M(GPS module)

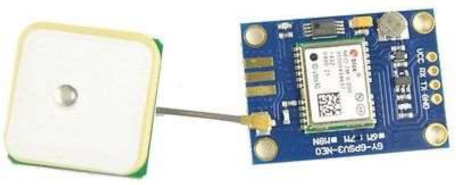
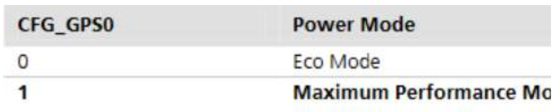


Figure 12. Neo-7M GPS module

Raspberry pi contains parallel i/o (input/output) communication terminals. What we're going to use this project is the Universal Asynchronous Receiver & Transmitter. Serial Bus (Serial Bus) is the process of continuously transmitting data in bits. RX (Data Received), TX (Data Transmission) terminals are prepared and serial communications are made to obtain information from the GPS module, where the acquired information is parsed to obtain latitude and longitude data.

The NEO-7M is a standalone GPS receiver that's used for navigation. The GPS receiver links with GPS satellites to get its own location. It, then, outputs the latitude and longitude of its position as serial data. The NEO-7 module is based on a 50-channel, ublox-6 positioning engine that boasts a time-to-first-fix (TTFF) of one second. This GPS engine has two million correlators and is capable of massive parallel time/frequency space searches. This enables it to find satellites instantly. The module also has a small form-factor that makes it ideal for battery-operated mobile devices. The NEO-7M GPS modem operates on a supply voltage of 2.7 to 3.6V. It communicates GPS data according to the NMEA or UBX protocol. While NMEA is a standard ASCII protocol, UBX is a u-blox proprietary binary protocol.

The receiver chipset has UART, USB, SPI, and DDC (I2C-compliant) interfaces to communicate this data. The chipset has three configuration pins — CFG-GPS0 — is used to enable boot configuration of the power mode.



CFG_GPS0	Power Mode
0	Eco Mode
1	Maximum Performance Mo

Figure 13. CFG_GPS0

The other two configuration pins — GFG_COM0 and GFG_COM1 — are used to decide whether GPS data is communicated by using the NMEA protocol or UBX protocol.

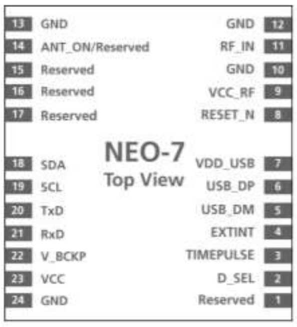


Figure 14. NEO-7 Pin Assignment

In the module used here, the NEO-6 modem is pre-configured to output data using the serial (UART) interface and encode GPS data to the NMEA protocol.

The configuration pins CFG_COM0 and CFG_COM1 are pulled to HIGH and, as a result, the GPS data is communicated over the NMEA protocol at a baud rate of 9600 bps. As you can see from the above table, with this configuration, the NMEA data includes GSV, RMC, GSA, GGA, GLL, and VTG messages.

The module only exposes four channels. The module has the following pin assignment:

Pin Name	Pin Description
VCC	Positive Supply Pin
RX	UART Receive Pin
TX	UART Transmit Pin
GND	Ground

Figure 15. Module pin assignment

The maximum navigation update rate of the module is 5 Hz. So, a controller or embedded computer can read the GPS data from the modem in a minimum of 0.2 seconds.

NMEA is an acronym for the National Marine Electronics Association. In context to a GPS, NMEA is a standard data format supported by all GPS manufacturers. It's a protocol for GPS data that is used by the GPS receivers and associated software.

NMEA-formatted GPS data can be transmitted over a variety of data communication standards, including UART, SPI, I2C, USB, Wi-Fi, UHF, and Bluetooth.

In NMEA protocol, the GPS data is communicated as NMEA message strings. There are also GPS receivers with different capabilities. According to their capabilities, they typically communicate a subset of NMEA message strings. All NMEA messages start with the $ character, followed by the message ID and various data fields that are separated by a comma. The message ends with an asterisk (*), followed by checksum character. Lastly, a carriage return, and line feed serve as the end characters. The GPS module used here communicates GSV, RMC, GSA, GGA, GLL, and, VTG NMEA messages. The message we need to receive here is GPGGA.

GPGGA – indicates the time, position, and fix related data. It has this format:

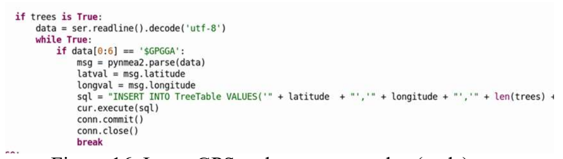
```
$GPGGA,hhmmss:ss,Latitude,N,Longitude,E,FS,NoSV,HDOP,msl,m,A
ltref,m,DiffAge,DiffStation*cs<CR><LF>
```

Figure 12. GPGGA

Now, We are ready to receive the necessary information from the GPS module.

- Install MariaDB on Raspberry Pi

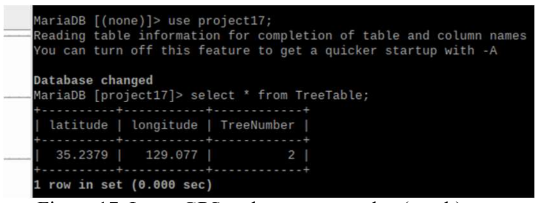
For store GPS and tree amount data to SQL, this paper use MariaDB. This database stores GPS and tree amount data. Insert GPS and tree amount data conducted with python code.



```python
if trees is True:
    data = ser.readline().decode('utf-8')
    while True:
        if data[0:6] == '$GPGGA':
            msg = pynmea2.parse(data)
            latval = msg.latitude
            longval = msg.longitude
            sql = "INSERT INTO TreeTable VALUES('" + latitude  + "','" + longitude + "','" + len(trees) +
            cur.execute(sql)
            conn.commit()
            conn.close()
            break
```

Figure 16. Insert GPS and tree amount data(code)

When Raspberry Pi detects a tree, 'Figure 6' calls GPS data. And the parsed longitude and latitude value will call SQL saved to DB that we have defined in advance. Then, if each interval has a tree, it will insert a row containing the latitude and longitude trees.



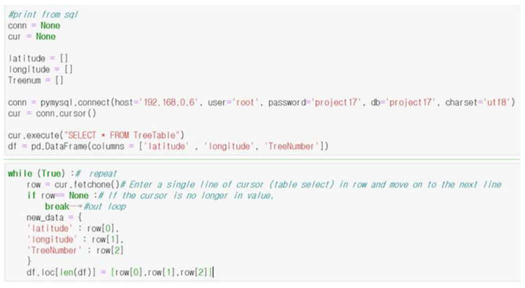
```
MariaDB [(none)]> use project17;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [project17]> select * from TreeTable;
+----------+-----------+------------+
| latitude | longitude | TreeNumber |
+----------+-----------+------------+
|  35.2379 |   129.077 |          2 |
+----------+-----------+------------+
1 row in set (0.000 sec)
```

Figure 17. Insert GPS and tree amount data(result)

This process will stack the data in the MariaDB of Raspberry Pi. Then it is possible to access GPS and tree amount data remotely. Finally, extract valuable data from MariaDB and visualize this data with a map.

Raspberry Pi is a device to collect data. Using an external computer to output data would be better because it is easy to receive the results and more convenient to process.

We build up an internal network. First, we set up Raspberry Pi to connect to a fixed internal network. When this device collects data and returns to a base that has access to the internal network, the following actions begin:



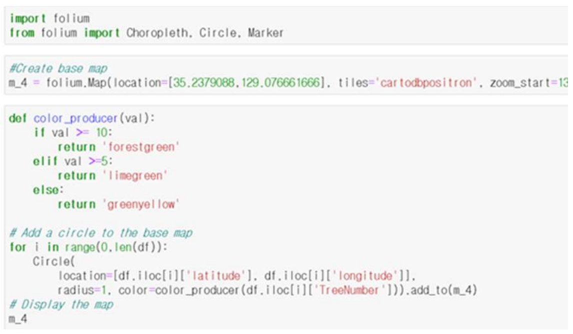
```
#print from sql
conn = None
cur = None

latitude = []
longitude = []
Treenum = []

conn = pymysql.connect(host='192.168.0.6', user='root', password='project17', db='project17', charset='utf8')
cur = conn.cursor()

cur.execute("SELECT * FROM TreeTable")
df = pd.DataFrame(columns = ['latitude' , 'longitude', 'TreeNumber'])
```

```
while (True) :#  repeat
    row = cur.fetchone()# Enter a single line of cursor (table select) in row and move on to the next line
    if row== None :# If the cursor is no longer in value.
        break→#out loop
    new_data = {
    'latitude' : row[0],
    'longitude' : row[1],
    'TreeNumber' : row[2]
    }
    df.loc[len(df)] = [row[0],row[1],row[2]]
```

Figure 18. Extract data remotely

External computers can access Raspberry Pi DB with fixed IP. However, external computers only have access to the DB but cannot modify the data, which prevents data contamination.

The external computer connects to the DB calling the data row by row from that table. We reconstruct the data in Dataframe format to easily manipulate this data. Now, we have data on latitude, longitude, and the number of trees.



```
import folium
from folium import Choropleth, Circle, Marker
```

```
#Create base map
m_4 = folium.Map(location=[35.2379088,129.076661666], tiles='cartodbpositron', zoom_start=13
```

```
def color_producer(val):
    if val >= 10:
        return 'forestgreen'
    elif val >=5:
        return 'limegreen'
    else:
        return 'greenyellow'

# Add a circle to the base map
for i in range(0,len(df)):
    Circle(
        location=[df.iloc[i]['latitude'], df.iloc[i]['longitude']],
        radius=1, color=color_producer(df.iloc[i]['TreeNumber'])).add_to(m_4)
# Display the map
m_4
```

Figure 19. Visualize data remotely (code)

We're going to visualize them. Python's representative visualization library, 'folium'[11], will be used to mark the location of the tree. We create the base map and a function that determines the circle's color displayed on the map. The more trees converge, the deeper the green color is. Repeating marking practices by the length of the data frame, we draw circles with different green color tones depending on the number of trees.

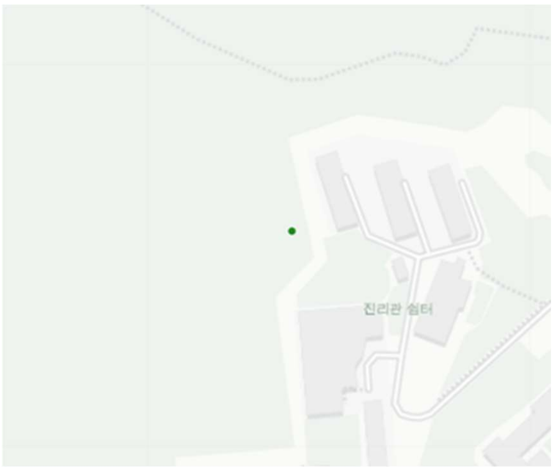


Figure 17. Visualize data remotely (result)

After all, data extraction, storage, and visualization have been completed.

In summary, we visualize the picture and video that a drone takes in a particular area. We can identify where trees are concentrated depending on the location and color of the circle stamp.

D. Conclusion

This paper examines tree detection and visualizing its distribution. Examination use 'Haar Cascade', Raspberry pi. 'Haar Cascade' uses for detecting tree and Raspberry pi uses for sending tree data remotely.

This paper can find the possibility of detecting tree and visualizing it to map. Some result shows that HD content has more accuracy than others. And because of the limitation of the Raspberry pi, the detect speed is slow. It means that 'Haar Cascade' is maybe not proper to 4K images. But if enough data and machine available, then this examination could improve and can uses for many other fields like tracking tree destroy of forest.

References

[1] P. Volia and M. Jones. Rapid obejct detection using a boosted cascade of simple features, IEEE, 2003

[2] https://www.bgci.org/our-work/projects-and-case-studies/global-tree-assessment

[3] Shang, X. & Chisholm, L. A. Classification of Australian native forest species using hyperspectral remote sensing and machine-learning classification algorithms. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 7, 2481–2489, 2014

[4] Boschetti, M., Boschetti, L., Oliveri, S., Casati, L. & Canova, I. Tree species mapping with Airborne hyper-spectral MIVIS data: The Ticino Park study case. Int. J. Remote Sens. 28, 1251–1261, 2007

[5] Jansson, G. & Angelstam, P. Threshold levels of habitat composition for the presence of the long-tailed tit (Aegithaloscaudatus) in a boreal landscape. Landsc. Ecol. 14, 283–290, 1999

[6] European Environmental Agency. European Forest Types. Categories and Types for Sustainable Forest Management Reporting and Policy. EEA Technical Report No9, 2006

https://www.eea.europa.eu/publications/technical_report_2006_9, https://doi.org/10.3832/efor0425-003

[7] Masanori Onishi & Takeshi Ise, Explainable identification and mapping of trees using UAB RGB image and deep learning, Scientific Reports 11, Article number:903 ,2021

[8] Fassnacht, F. E. et al. Review of studies on tree species classification from remotely sensed data. Remote Sens. Environ. 186, 64–87, 2016

[9] Sasalak Tongkaw & Aumnat Tongkaw, A Comparison of Database Performance of MariaDB and MySQL with OLTP Workload, IEEE, 2016

[10] J. Canny. A Computational Approach to Edge Detection, IEEE, 1986

[11] http://python-visualization.github.io/folium/