UAV Ground Detection

Tree detection and tracking change species and amount of tree

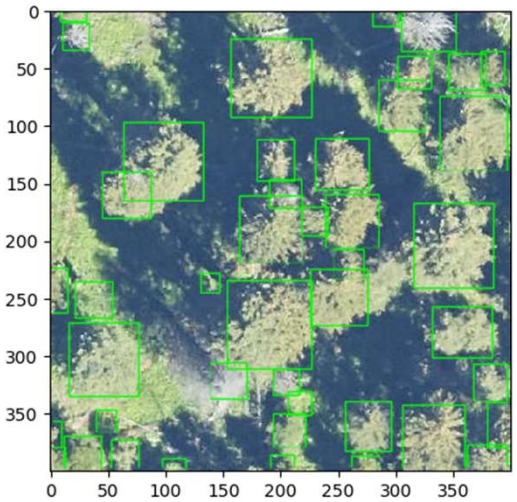
Christian Ekeigwe, Daehyeon Jeong, Jaeyeong Shim, Jeonghwan Kang, Seoungheong Jeong

Project 17

PURDUE UNIVERSITY | Polytechnic Institute

Detecting Tree model

Deep Forest



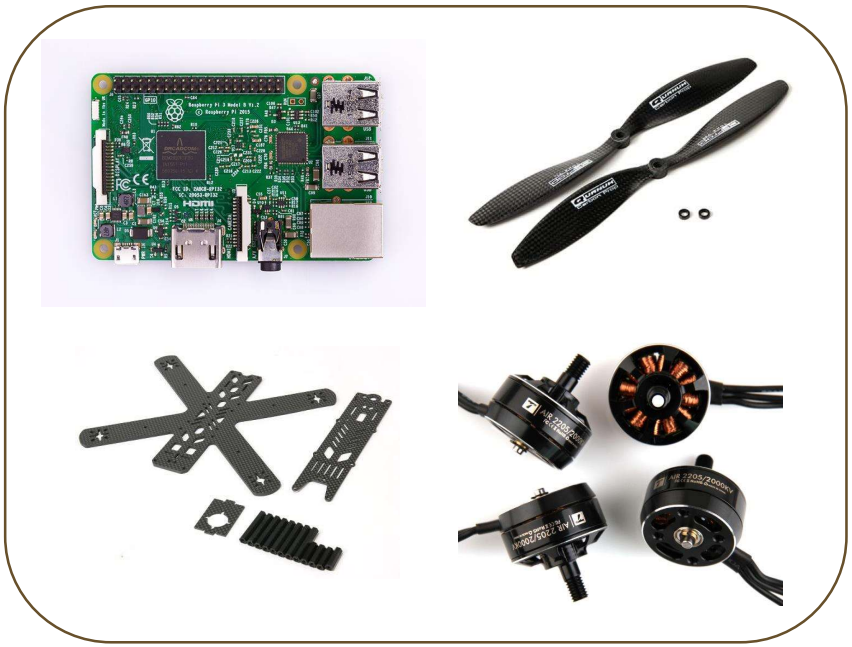
https://deepforest.readthedocs.io/en/latest/landing.html

- Training and predicting individual tree airborne RGB image

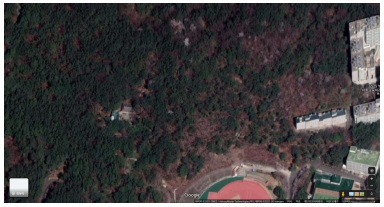
Collecting data using UAV

- When using UAV ground detecting we need to use real time object detection

Solution



Building drones



Google Maps Platform

UCI Machine Learning Repository

kaggle

Covertype Data Set

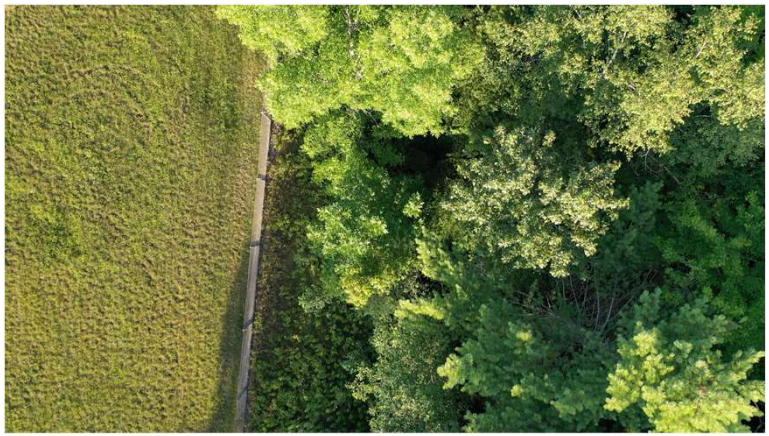
Download: Data Folder, Data Set Description

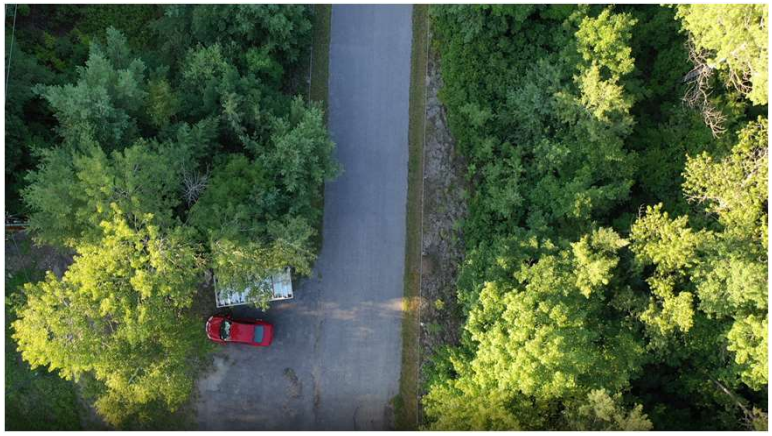
Abstract: Forest CoverType dataset

Data Set Characteristics:	Multivariate	**Number of Instances:**	581012	**Area:**	Life
Attribute Characteristics:	Categorical, Integer	**Number of Attributes:**	54	**Date Donated**	1998-08-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	334249

Collecting external data

Data



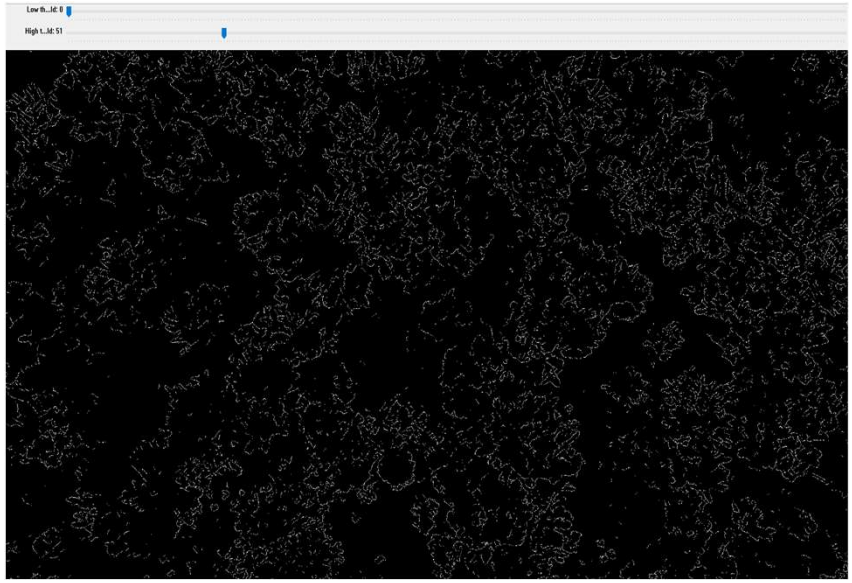
The video capture shot we get from Purdue

PURDUE UNIVERSITY | Polytechnic Institute

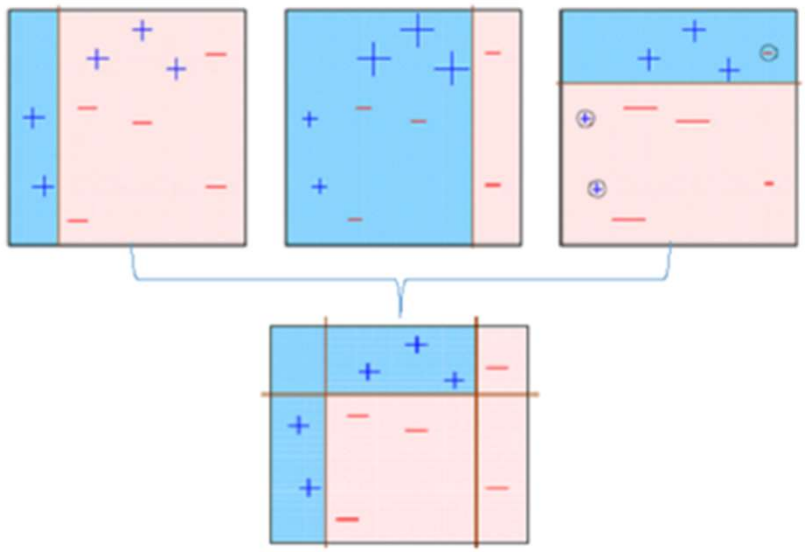
Plan

Step 1	Get data from video	Make modeling system
Step 2	Classify data (Distinguish Tree Species)	Modeling data
Step 3	Real time receive data from UAV	Add/Del data from GEO chart

Step 1_Detecting tree



Edge detection

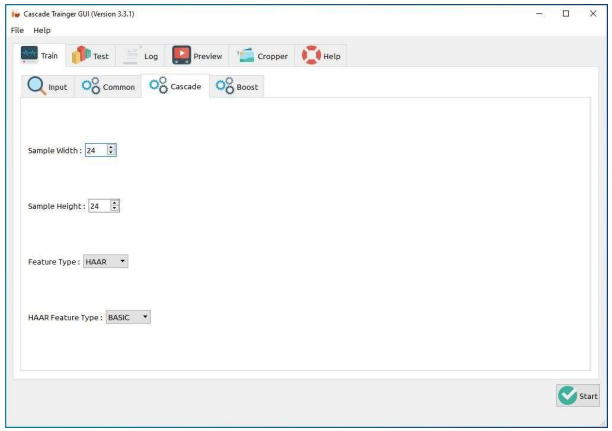


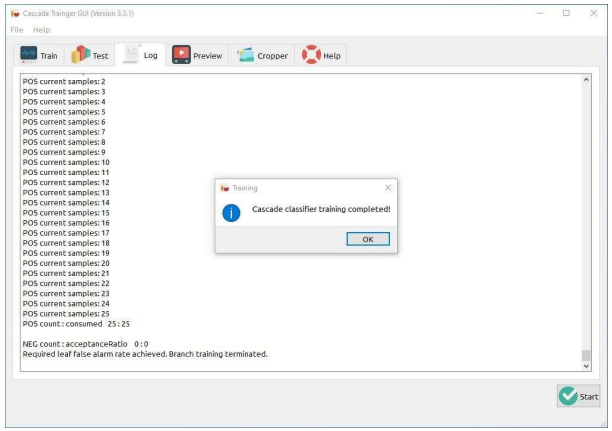
Haar Cascades

Classify Data

```python
import os
import re
import time
import cv2
import numpy as np
from os.path import isfile, join

tree_classifier = cv2.CascadeClassifier('<Cascade_File_Path>')

cap = cv2.VideoCapture('<Video_File_Path>')

while True:
    time.sleep(.05)
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    trees = tree_classifier.detectMultiScale(gray, 1.3, 5)
    for (x, y, w, h) in trees:
        image = cv2.rectangle(frame, (x, y), (x+w, y+h), (0,0,255), 2)
        cv2.imshow('Trees', image)
        #cv2.namedWindow('Trees', cv2.WINDOW_NORMAL) #optional
        #cv2.resizeWindow('Trees', 1900, 1000) #optional
        cv2.waitKey(1)

cap.release()
cv2.destroyAllWindows()
```

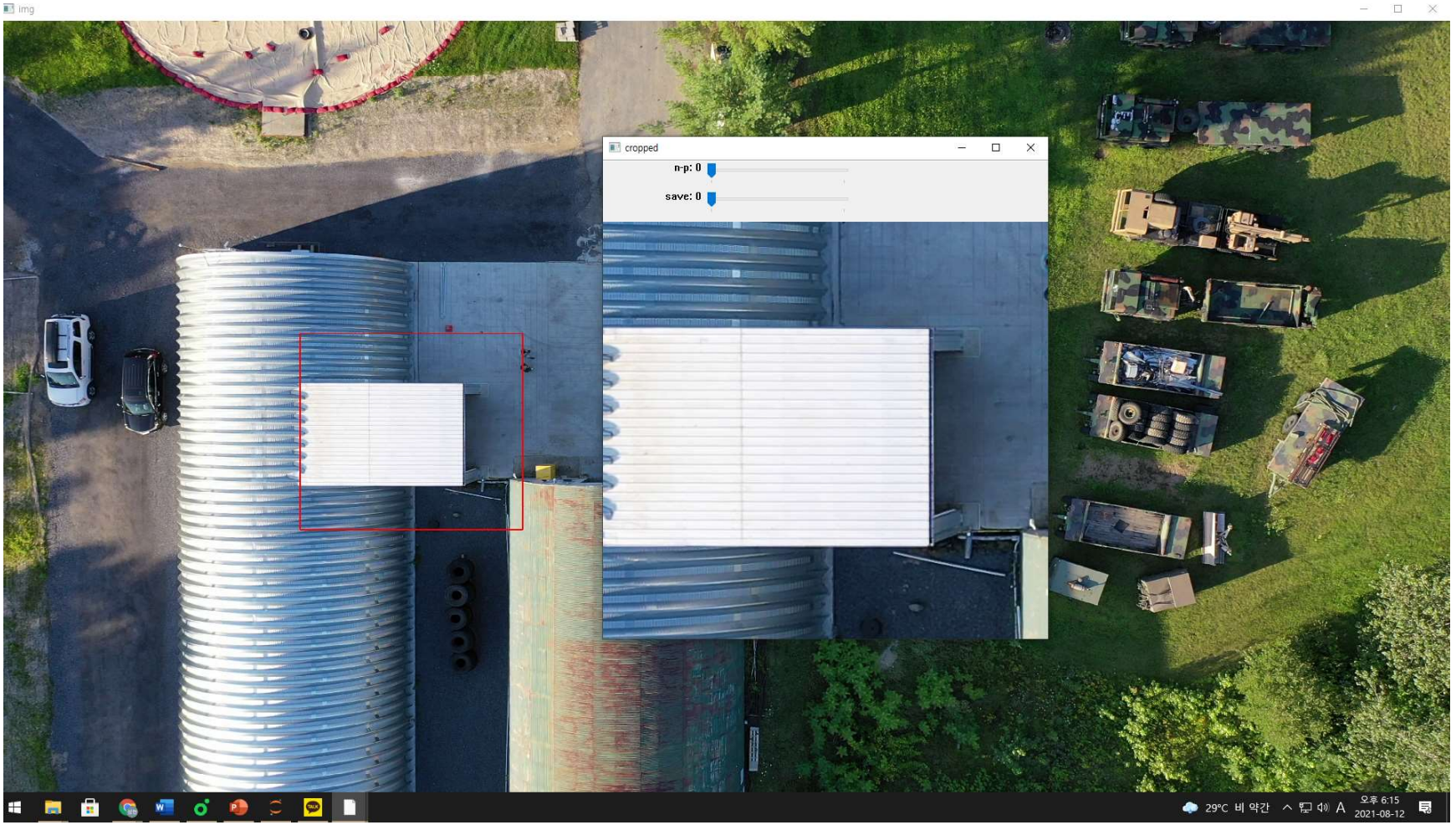
Detecting images from video





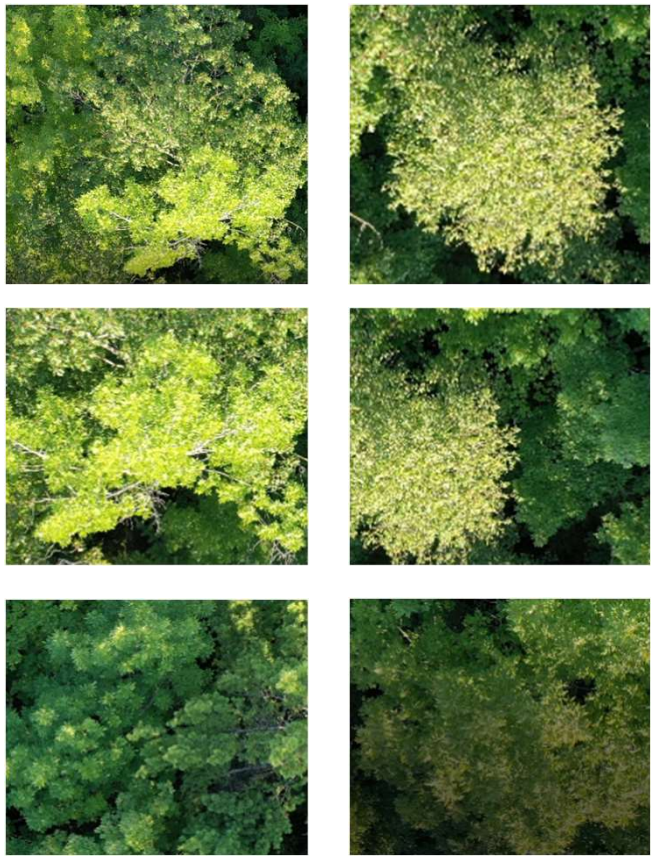
CascadeTrainer

Making data set

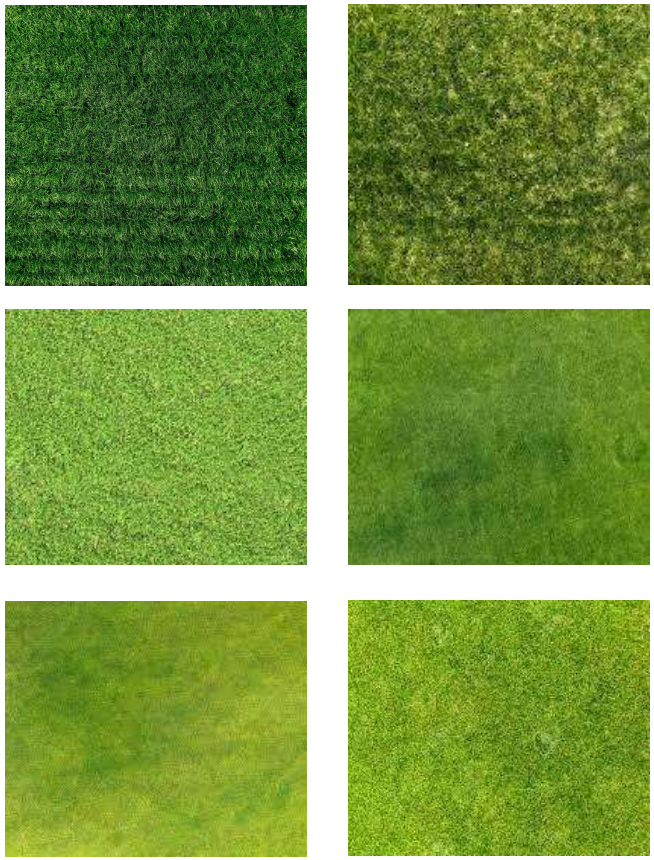


Making data set

Image Labeling



Positive



Negative

PURDUE UNIVERSITY | Polytechnic Institute

Operating System

1 Step : detecting from Raspberry Pi

2 Step : Store to SQL

3 Step : Extraction & Visualization

1 Step : detecting from Raspberry Pi

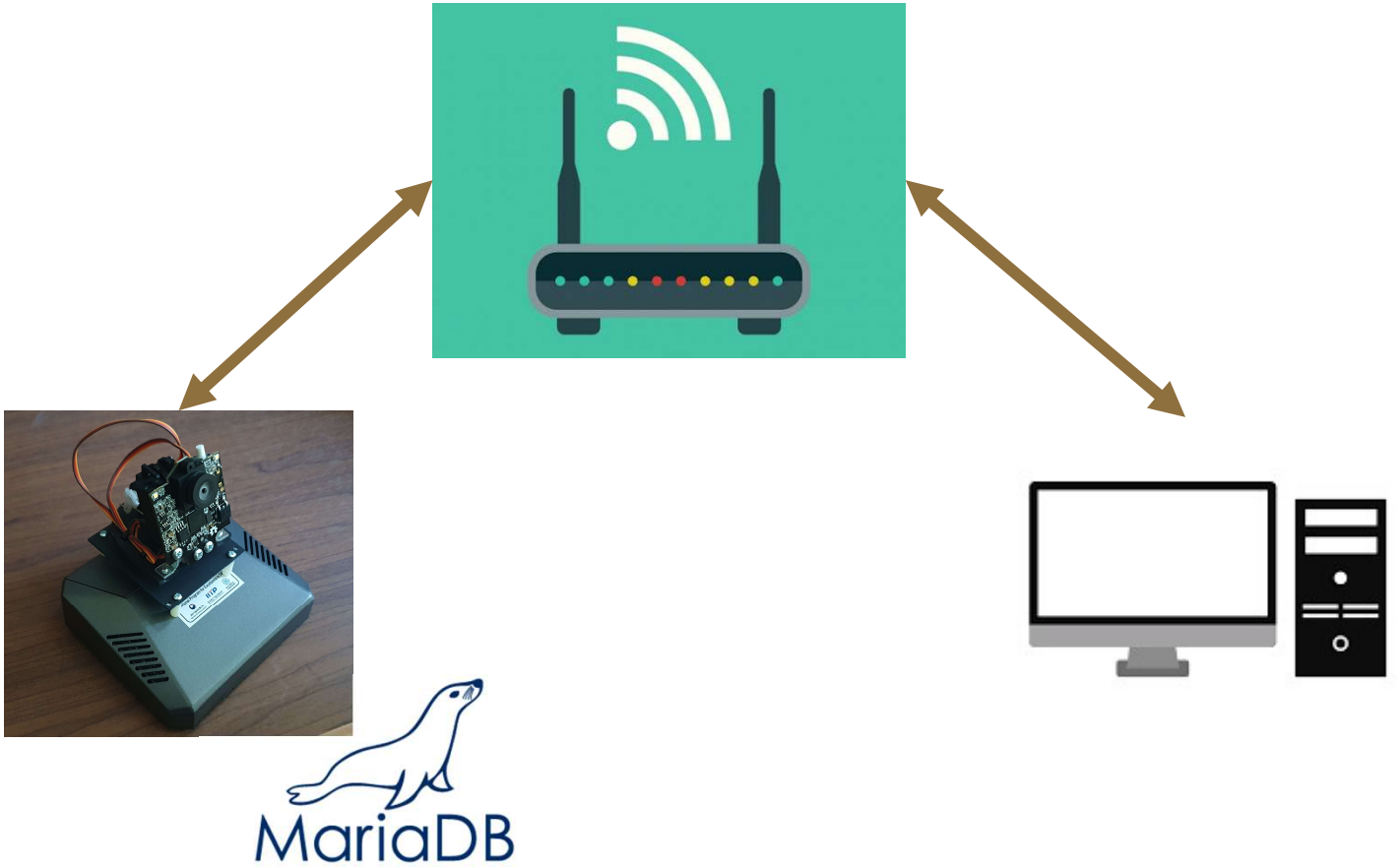
1	2	3
4	5	6
7	8	9

2 Step : Store to SQL


```
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)
```

```
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
```

3 Step : Extraction & Visualization



3 Step : Extraction & Visualization

```
#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]]
```

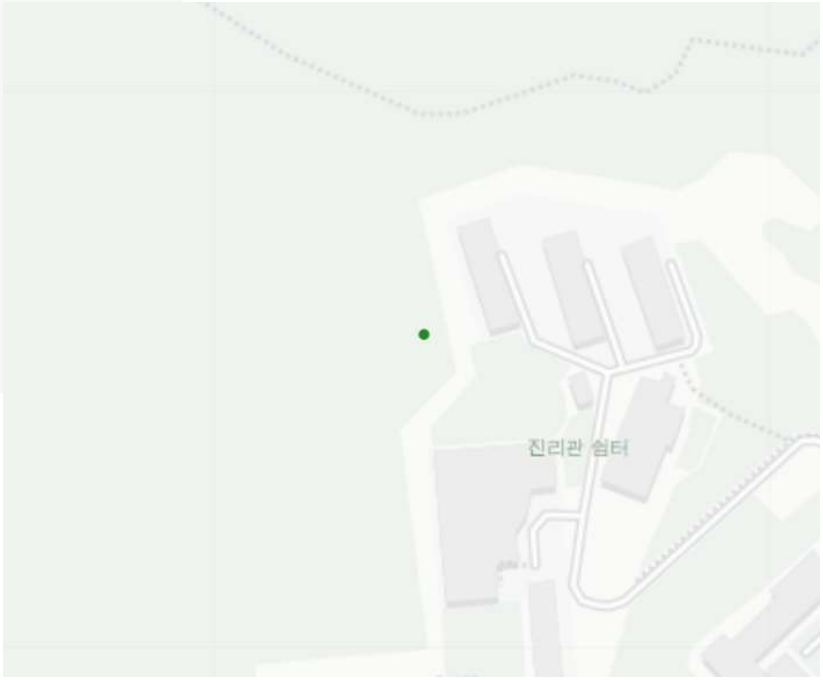
	latitude	longitude	TreeNumber
0	35.237909	129.076662	2.0

3 Step : Extraction & Visualization

```
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
```



Thank you

Questions?

polytechnic.purdue.edu

/ TechPurdue

PURDUE UNIVERSITY | Polytechnic Institute