Photometery using automated drone swarming

~By Ashutosh Suthar

Mentored by Prasannam Kumar Sah

<u>**Objective**</u> :-

- to make 3-d model using images captured by drone swarming

<u>**Tools used**</u> :-

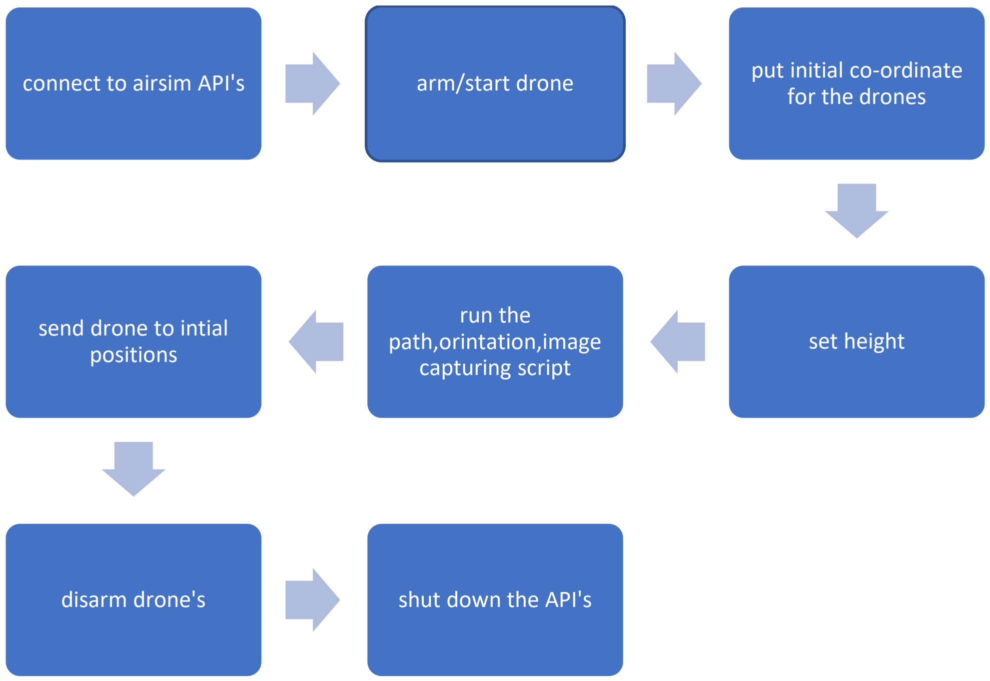
- airsim
- unreal open source environment(nh) for airsim
- python(for script)

<u>**Path planning**</u> :-

- path used in my simulation is a circle around a central body (whose 3-d model is to be generated).Drone's are moved on circular path by using coordinate form of a circle, where x-co-ordinate ,and y co-ordinate are $(x_1 + R\cos\theta\,, y_1 + R\sin\theta)$, where (x_1, y_1) is the co-ordinate of the center.

<u>**Steps to write python script :-**</u>

1)


connect to airsim API's
arm/start drone
put initial co-ordinate for the drones
set height
run the path,orintation,image capturing script
send drone to intial positions
disarm drone's
shut down the API's

<u>Steps to proceed the project</u> :-

1. Check is all the programs are running well.
2. Write the script for defining the path of drones, and write the script in settings.json to introduce the drone to the environment.
3. Identify the object in the simulation, run test trials to identify the dimensions of its area.
4. Make necessary change to the python script to introduce correct dimensions of the area.
5. Run the script to take images.
6. Align all the photos taken to identify points in metashapes(or any similar software).
7. Build mesh to generate 3-d model of the object.

<u>Problems encountered :-</u>

- Collisions with objects in the environment.
  Solution:-increase height and set orientation.

<u>MY settings.json script :-</u>

```
{
    "SeeDocsAt": "https://github.com/Microsoft/AirSim/blob/master/docs/settings.m
d",
    "SettingsVersion": 1.2,
    "SimMode":"Multirotor",

    "Vehicles": {
          "drone6":{
            "VehicleType": "SimpleFlight",
            "X": 0,
            "Y": 2,
            "Z": -1,
            "DefaultVehicleState": "Disarmed",
            "EnableCollisionPassthrogh": false,
            "EnableCollisions": true,
            "AllowAPIAlways": true,
            "EnableTrace":false,
            "IsFpvVehicle":false,
            "RC": {
                "RemoteControlID": 0,
                "AllowAPIWhenDisconnected": false
            }
          },
          "drone7":{
            "VehicleType": "SimpleFlight",
            "X": 0,
            "Y": 4,
            "Z": -1,
            "DefaultVehicleState": "Disarmed",
            "EnableCollisionPassthrogh": false,
            "EnableCollisions": true,
            "AllowAPIAlways": true,
            "EnableTrace":false,
            "IsFpvVehicle":false,
            "RC": {
                "RemoteControlID": 0,
                "AllowAPIWhenDisconnected": false
            }
          }
    },
    "CameraDefaults": {
        "CaptureSettings": [
          {
            "ImageType": 0,
            "Width": 1280,
            "Height": 720,
            "FOV_Degrees": 75
          }
      ],
      "Gimbal": {
        "Stabilization": 1
      }
    }
  }
```

<u>My python script : -</u>

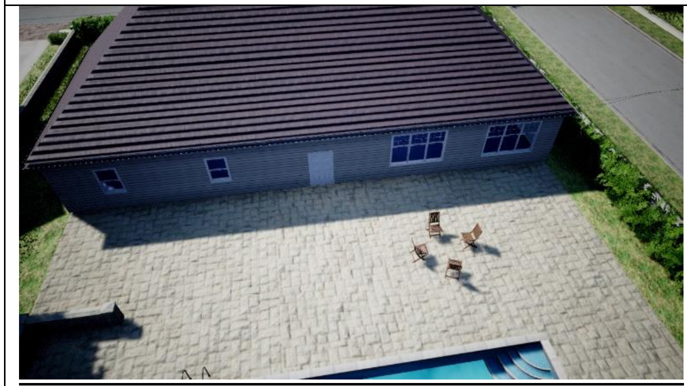
```
import math
import os
import time
import airsim
import numpy as np

#connecting airsim
client = airsim.MultirotorClient()
```

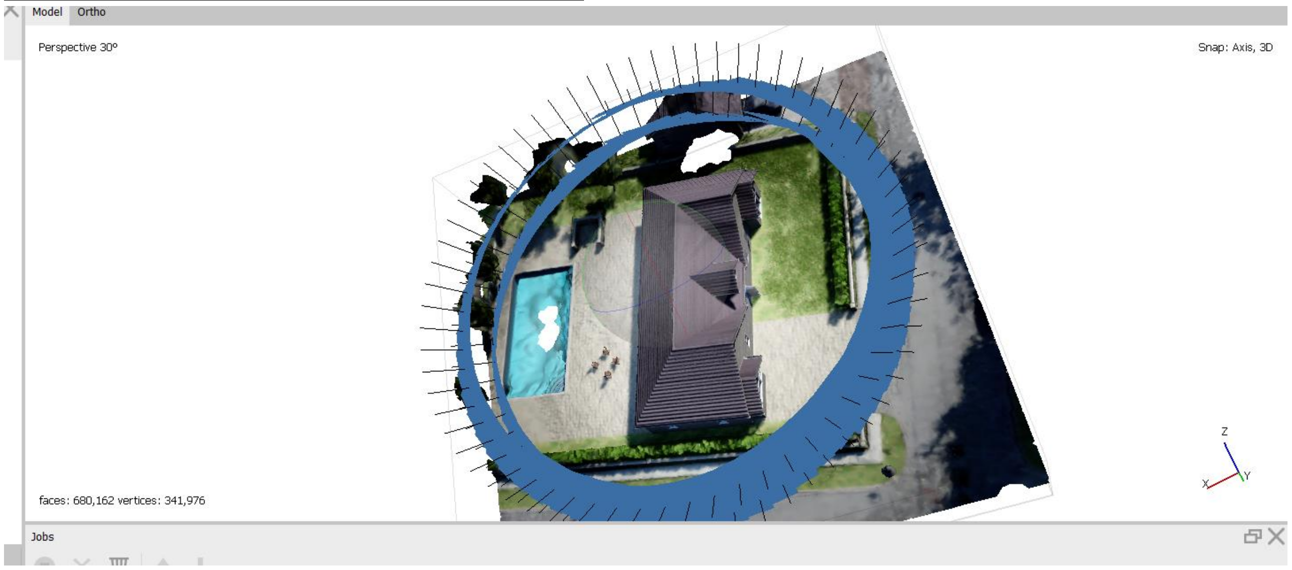
```
client.confirmConnection()
#connecting api's
client.enableApiControl(True,vehicle_name="drone6")
client.enableApiControl(True,vehicle_name="drone7")
#arming drone
client.armDisarm(True,vehicle_name="drone6")
client.armDisarm(True,vehicle_name="drone7")
#initial location setup
d6=client.takeoffAsync(vehicle_name="drone6")
d7=client.takeoffAsync(vehicle_name="drone7")
d6.join()
d7.join()
#setting heights
z6=-15
z7=-10
d6=client.moveToPositionAsync(0,0,z6,5,vehicle_name='drone6')
d6.join()
d6=client.moveToPositionAsync(0,0,z7,5,vehicle_name='drone7')
d7.join()
#Planing path
d = 3 #image taking differrence
s = 1 #survey drone speed
camera_pitch_angle6=-45 #camera pitch angle
camera_pitch_angle7=-30
radius=-22
k=1
theta=0
while(theta<=2*np.pi):
    x=radius-(radius*np.cos(theta))
    y=radius-(radius*np.sin(theta))
    #for camera pose defined function are
    #airsim.pose(position,orientation)
    #airpose.to_quaternion(pitch,roll,yaw)
    #client.simSetVehiclePose(pose of vehicle,Collision(bool),vehicle_name)
    #it sets a default setting for the position and orientatoin for camera
    d6=client.moveToPositionAsync(x,y-2,z6,s,vehicle_name='drone6')
    d6.join()
    d7=client.moveToPositionAsync(x,y-4,z7,s,vehicle_name='drone7')
    d7.join()
    vehicle_pose = airsim.Pose(airsim.Vector3r(x,y-
2,z6),airsim.to_quaternion(0,0,theta))
    client.simSetVehiclePose(vehicle_pose,False,vehicle_name="drone6")
    camera_pose = airsim.Pose(orientation_val=airsim.to_quaternion(math.radians(c
amera_pitch_angle6),0,np.pi+theta))
    client.simSetCameraPose(0,camera_pose,vehicle_name='drone6')
    vehicle_pose = airsim.Pose(airsim.Vector3r(x,y-
4,z7),airsim.to_quaternion(0,0,theta))
    client.simSetVehiclePose(vehicle_pose,False,vehicle_name="drone7")
    camera_pose = airsim.Pose(orientation_val=airsim.to_quaternion(math.radians(c
amera_pitch_angle7),0,np.pi+theta))
    client.simSetCameraPose(0,camera_pose,vehicle_name='drone7')
    print("Camera angle are adjusted for all Drone's")
    # take images now
    #function used is:-
    #simgetImages( *make request* ,vehicle name)
    #ImageRequest(camera name, image type, pixels as float(bool), compress(bool)
)
    print("Drones are Taking images ...")
    responces = client.simGetImages([airsim.ImageRequest("0", airsim.ImageType.Sc
ene, False, False)],vehicle_name='drone6')
    response = responces[0]
    #np.reshape :- Gives a new shape to an array without changing its data.
    img1d = np.fromstring(response.image_data_uint8, dtype=np.uint8)
    img_rgb = img1d.reshape(response.height, response.width, 3)
    #naming and saving obtained image
    filename = time.strftime("%Y%m%d-6-%H%M%S")
    airsim.write_png(os.path.normpath(filename + '.png'), img_rgb)
    responces = client.simGetImages([airsim.ImageRequest("0", airsim.ImageType.Sc
ene, False, False)],vehicle_name='drone7')
    response = responces[0]
    img1d = np.fromstring(response.image_data_uint8, dtype=np.uint8)
    img_rgb = img1d.reshape(response.height, response.width, 3)
    filename = time.strftime("%Y%m%d-7-%H%M%S")
    airsim.write_png(os.path.normpath(filename + '.png'), img_rgb)
    theta=theta+0.1
print("we have completed survey procss..")
d6=client.moveToPositionAsync(0,0,-3,2,vehicle_name='drone6')
d6.join()
d7=client.moveToPositionAsync(0,0,-3,2,vehicle_name='drone7')
d7.join()
time.sleep(2)
#send drones to initial position and land
print("disarming the drones...")
client.armDisarm(False,"drone6")
client.armDisarm(False,"drone7")

#shut down the API's
client.reset()
client.enableApiControl(False,"drone6")
client.enableApiControl(False,"drone7")
```

<u>Sample images captured by drone's :-</u>

Dense-mesh generated by the metashape :-



±

What can be done more :-

- Using object detection to avoid obstacle at lower heights.
- Moving drones in definitive pattern.
    - Example, High-tech drones by Intel at the Winter Olympics

- Using swarm to make 3-d model of interior of buildings, houses etc.
- Using swarm drones as intermediate junction point for temporary communication in remote locations.

- Interconnecting drones to transfer food, medicine and more to different place's in hospital, cities, lab's etc.
- Surveillance of farms, disaster prone areas, forests.
- Surveillance of tactically important location's in military.