FIRESENSE

HARVARD SUMMER SCHOOL PROJECT

By: Akash, Ashok, Hardiv, Pavan, and Thomas

Identifying the Problem

- One of the most prevalent causes of property damage across the United States and the world, wildfires are both destructive and extremely difficult to contain.
    - In 2020 there were 58,950 wildfires compared with 50,477 in 2019, according to the National Interagency Fire Center. About 10.1 million acres were burned in 2020, compared with 4.7 million acres in 2019.

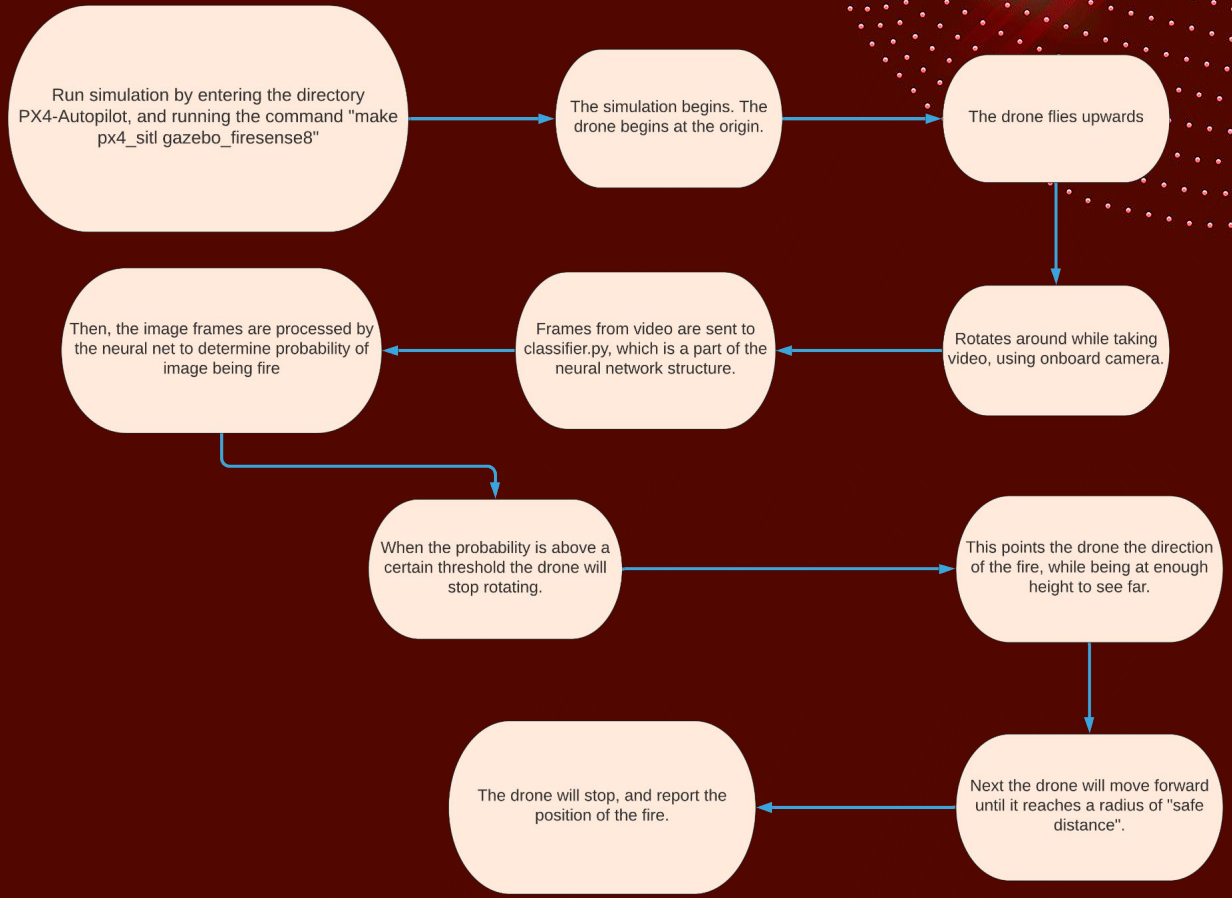
Identifying the Problem (Pt. 2)

- Lack of ability to detect wildfires when they’re smaller in size allows the problem to exacerbate itself, and eventually harm people and cause millions in property damage each year.
    - Having something that can cover vast regions to perform preliminary fire detection in search of smaller fires could help firefighters neutralize fires in their early stages, and thus minimize the potential harm to property, wildlife and people in the area.

Our Approach

- We wanted to create a vehicle that would facilitate the existence of the conditions put forth in the previous slide, conditions that would allow for firefighters to neutralize fires early.
    - Something that could cover vast regions, as well as efficiently perform fire detection.
- Thus, we decided on a drone (a small vehicle that can cover long distances in the air), equipped with sensors and neural networks. This would allow for fire detection to be efficiently performed from the air. Furthermore, the ability of the drones to cover large regions would increase the chance of a drone being able to identify a small fire.

Flowchart for Drone's Plan of Action


Run simulation by entering the directory PX4-Autopilot, and running the command "make px4_sitl gazebo_firesense8"
The simulation begins. The drone begins at the origin.
The drone flies upwards
Rotates around while taking video, using onboard camera.
Frames from video are sent to classifier.py, which is a part of the neural network structure.
Then, the image frames are processed by the neural net to determine probability of image being fire
When the probability is above a certain threshold the drone will stop rotating.
This points the drone the direction of the fire, while being at enough height to see far.
Next the drone will move forward until it reaches a radius of "safe distance".
The drone will stop, and report the position of the fire.

Softwares Used

- Gazebo 11 and 9
- Ubuntu 20.04.2 LTS
- PX4
- Python 3.8
- Atom
- Pycharm
- Github Desktop/Web
- Discord
- Unity
- VSCode
- C #
- MavSdk








01

Gazebo

- Preparing world
- Fixing Errors
- Integration with Neural net/drone

02

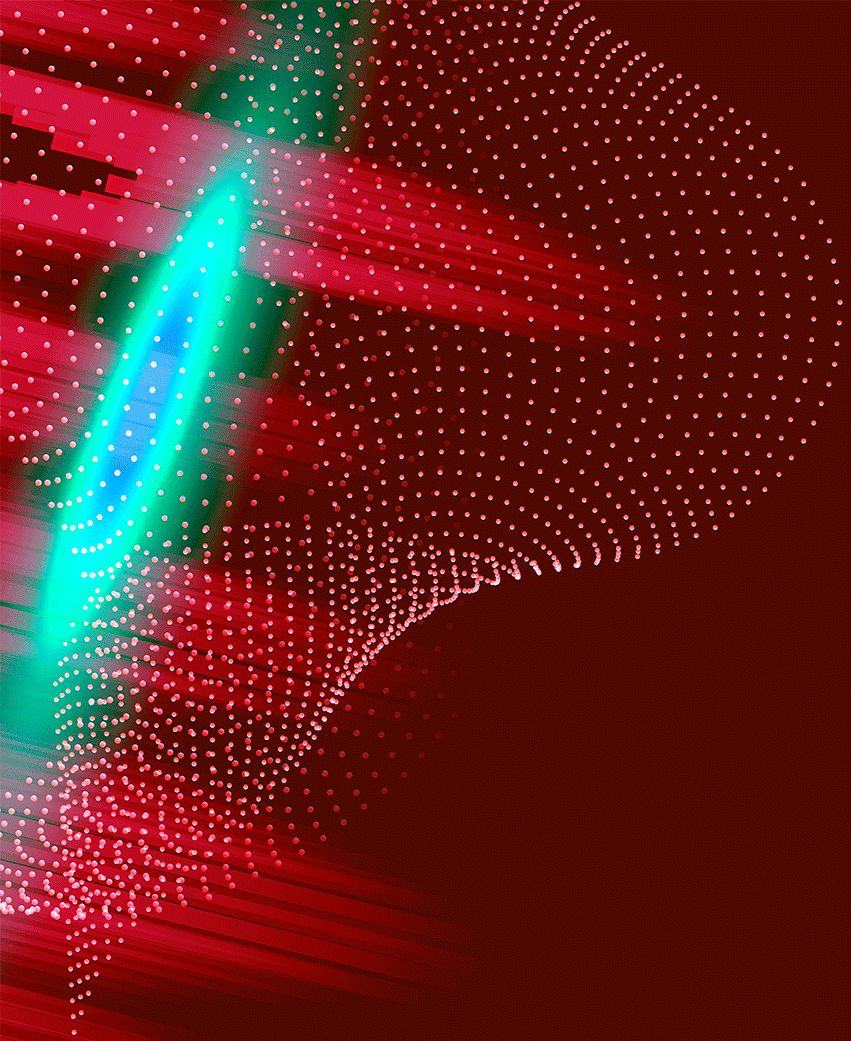
Neural Net

- 3 Convolutional Layers with Pooling
- Dropout Layer
- 2 Hidden Layers

03

Drone

- PX4-Typhoon H480
- Camera Feed
- Neural Net Integration
- Swarm



Gazebo

Our Simulator of Choice

Discussing Gazebo

- Gazebo was our simulator, visualizer, and modeler of choice.
    - Gazebo used in class, ROS, functionality
- On Gazebo we modeled the trees, and the general terrain to reflect that we wanted a forest-type environment for our final simulation.
    - Furthermore, to represent our fire, we used small oblong objects wrapped with a fire .png texture. We then placed those small oblong objects on the trees to represent our burning trees.
- The drone runs on PX4, and has been imported into Gazebo
    - Specifically it is PX4 Typhoon H480

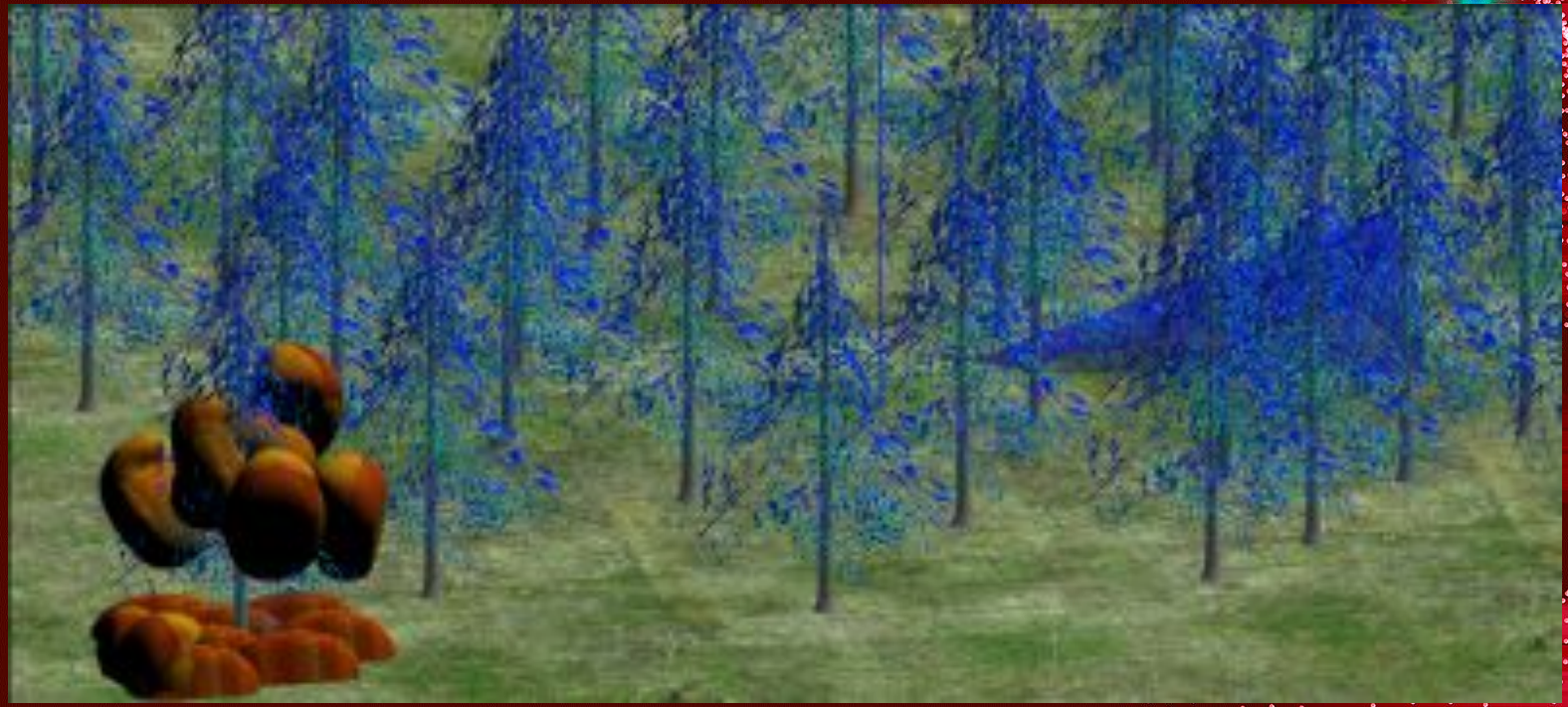
World Setup/Fires

- Download/install Gazebo with documentation provided in Github:

  https://github.com/Team-FireSense/FireSense/commit/4c6fdfe59602b35916232c5bda9a887a23b817cf
- Gazebo, PX4, and MAVSDK were used
- Difficulties producing Mesh fire texture
  - Used static instead of dynamic

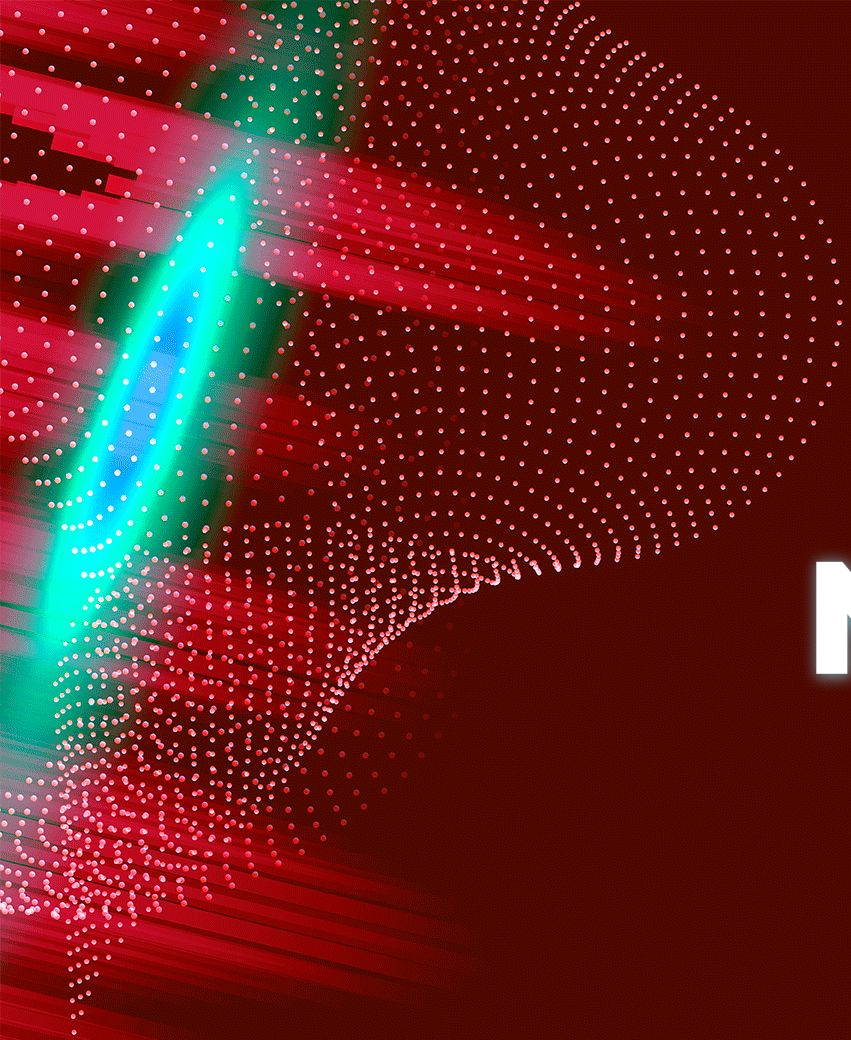
Progress In World Development (Implementation of Burning Tree Models)

Image Of Progress



Issues regarding Gazebo (and how they were Overcome)

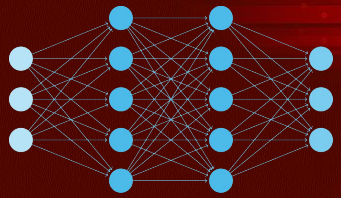
- File Save Errors - ex. Multiplying drone overlaying
- Stopped us from running PX4 Typhoon H480
    - Neural net integration not compatible
- Gazebo World save issue - root cause
    - Resolved through manual save overwrite
- Some computers would not render models properly.



Neural Net

The “Sense” in FireSense

How a Neural Network Works? (Generally)



- A neural network is a system inspired by the function of the human brain, specifically its capacity to learn, and make neural pathways
    - The general structure of a neural network consists of an input layer, a hidden layer, and the output layer.
- “A complex definition would be that a neural network is a computational model that has a network architecture. This architecture is made up of artificial neurons. This structure has specific parameters through which one can modify it for performing certain tasks.”

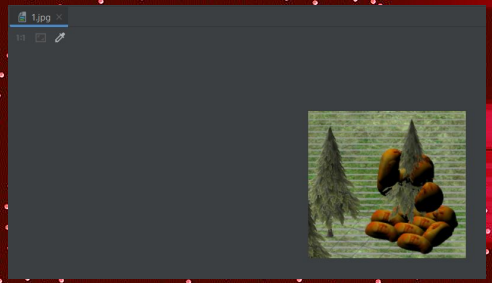
Step-by-Step for a Neural Network

1. Information is provided to the input layer, conveyed to the hidden layer.
2. Connections between the two layers assign weights to each point of info.
3. Bias is added to every input after the weights are multiplied with them individually.
4. The sum of weights is sent to the activation function.
5. Activation function makes decision on which nodes to be fired for extraction of a feature.
6. Application function applied to output layer to deliver output.
7. Weights adjusted, and output is reported and compared to original result to ‘learn’ and minimize errors in subsequent executions.

How our Neural Network Functions in Context of Fire Detection

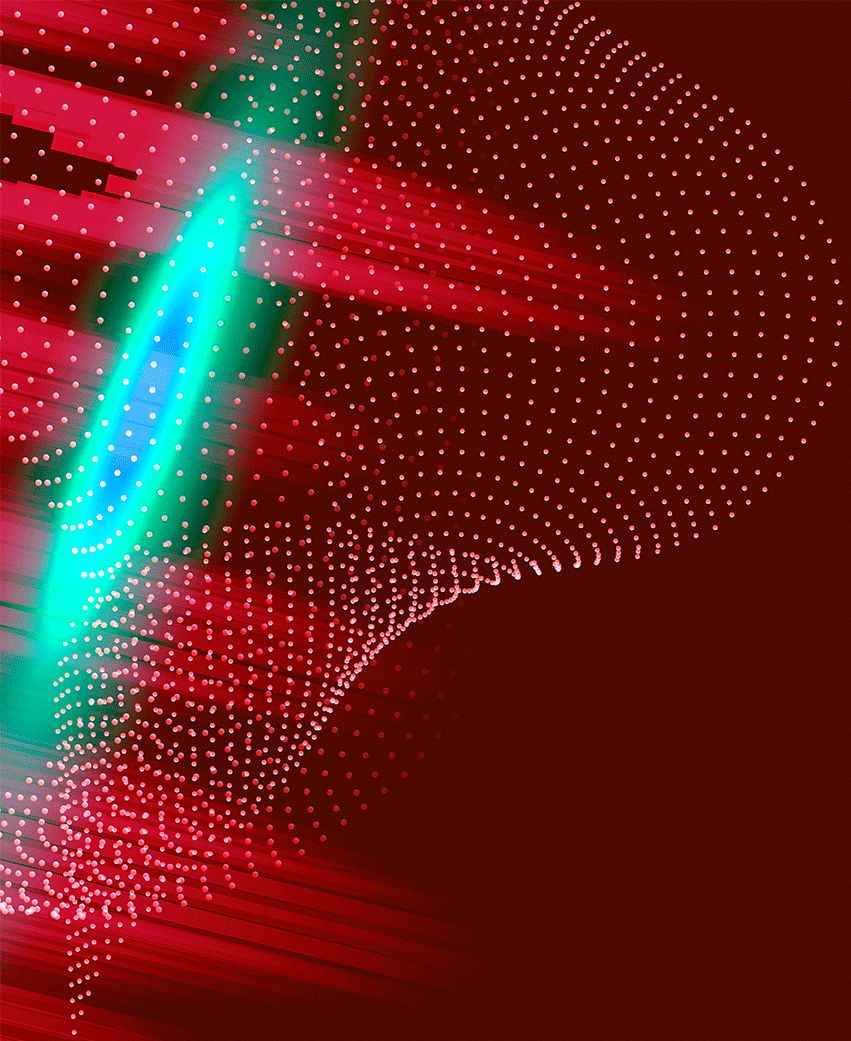
- In order to determine whether a given image represented a wildfire or not, we constructed and trained a neural network model on a dataset of 755 fire images and 244 non fire images.
- We constructed our neural network using the TensorFlow library, where we added 3 layers of convolution and pooling before a dropout layer and then flattened these into the input layer.
- This was then passed onto 2 hidden layers which then linked to the output layer, which had two nodes, one for the 'fire' classification and one for the 'non fire' classification

```
Enter file path of image to classify: C:\_code\python\FireSense\assets\testing\fire_images\1.jpg
2021-08-05 23:27:36.852807: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of
This image is most likely a fire.
--------------------------------
```



Issues Regarding Neural Network Integration (and How They Were Overcome)

- We had an issue regarding accuracy (65% to be exact) in the output of our neural network, and thus, some tweaks were needed to boost the accuracy.
    - An extra hidden layer was added to the neural network structure, which boost the accuracy. (93% accuracy)
    - Adding a dropout layer increased the accuracy to 96%.
- Increasing the input dataset also gave more opportunities for the neural network to "learn".

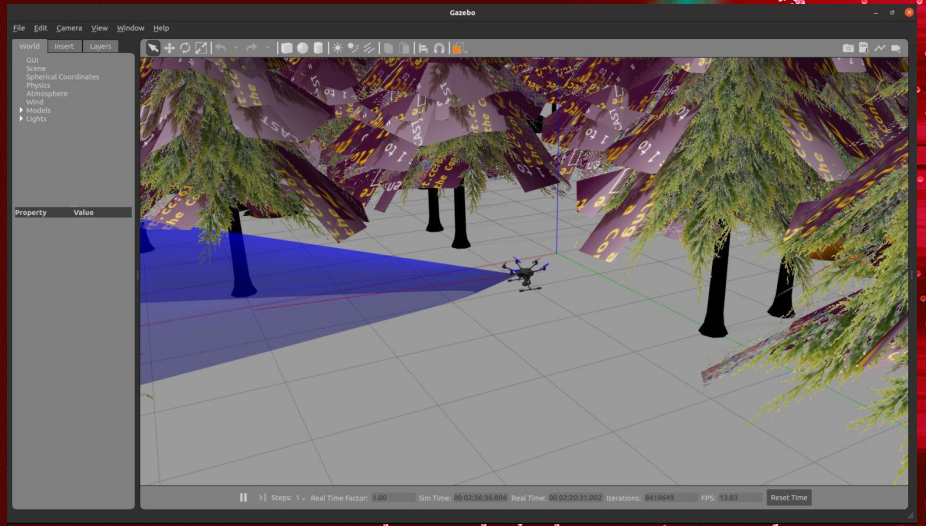


Drone

PX4 Typhoon H480

Discussion of our Drone

- The model of the drone: PX4-Typhoon H480
- Some sensors that the drone is equipped with:
  - Gyroscope
  - Camera
  - Barometer
  - Originally, we had the PX4-Iris in mind, however the Typhoon had better compatibility with our camera sensors

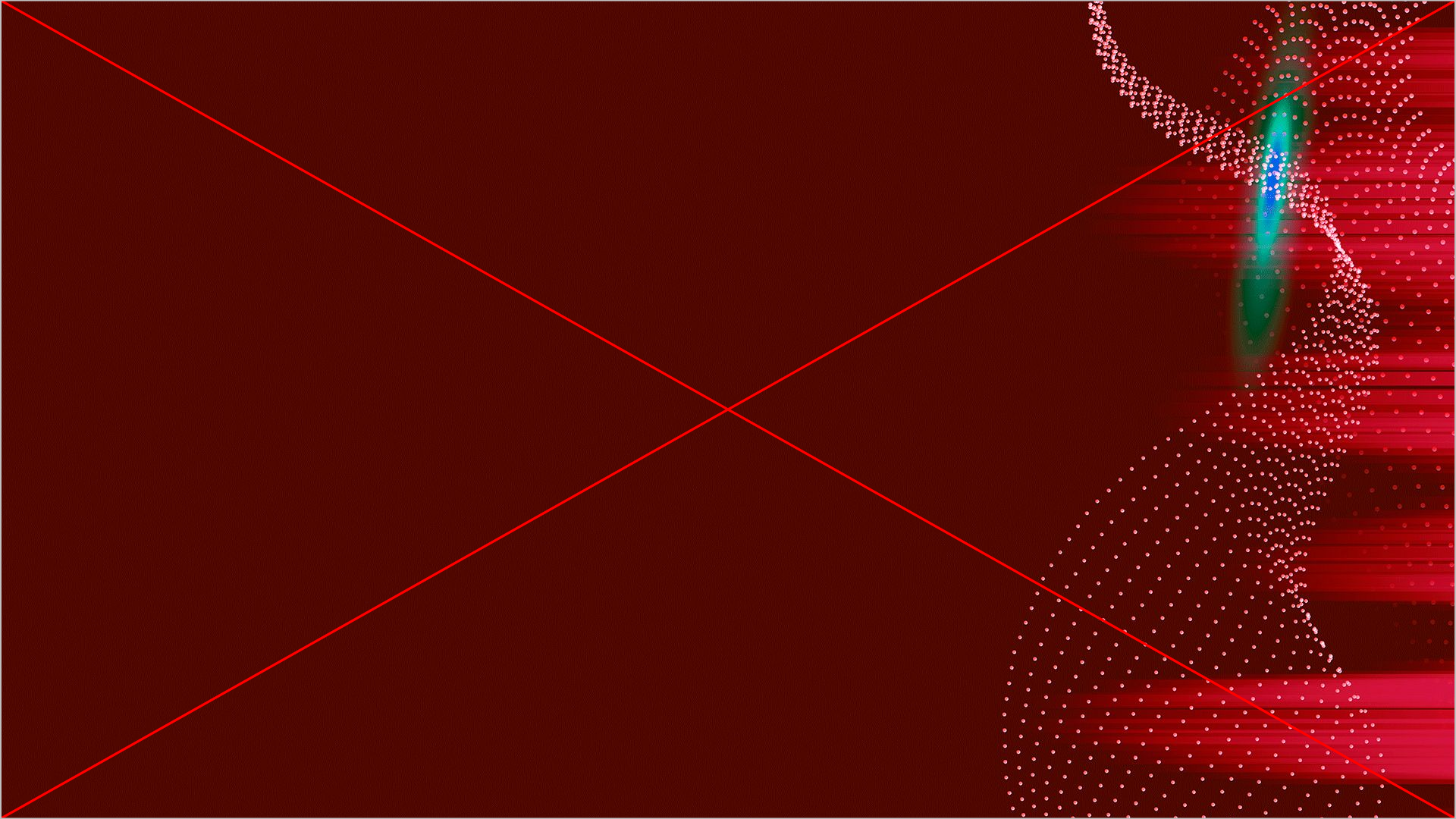


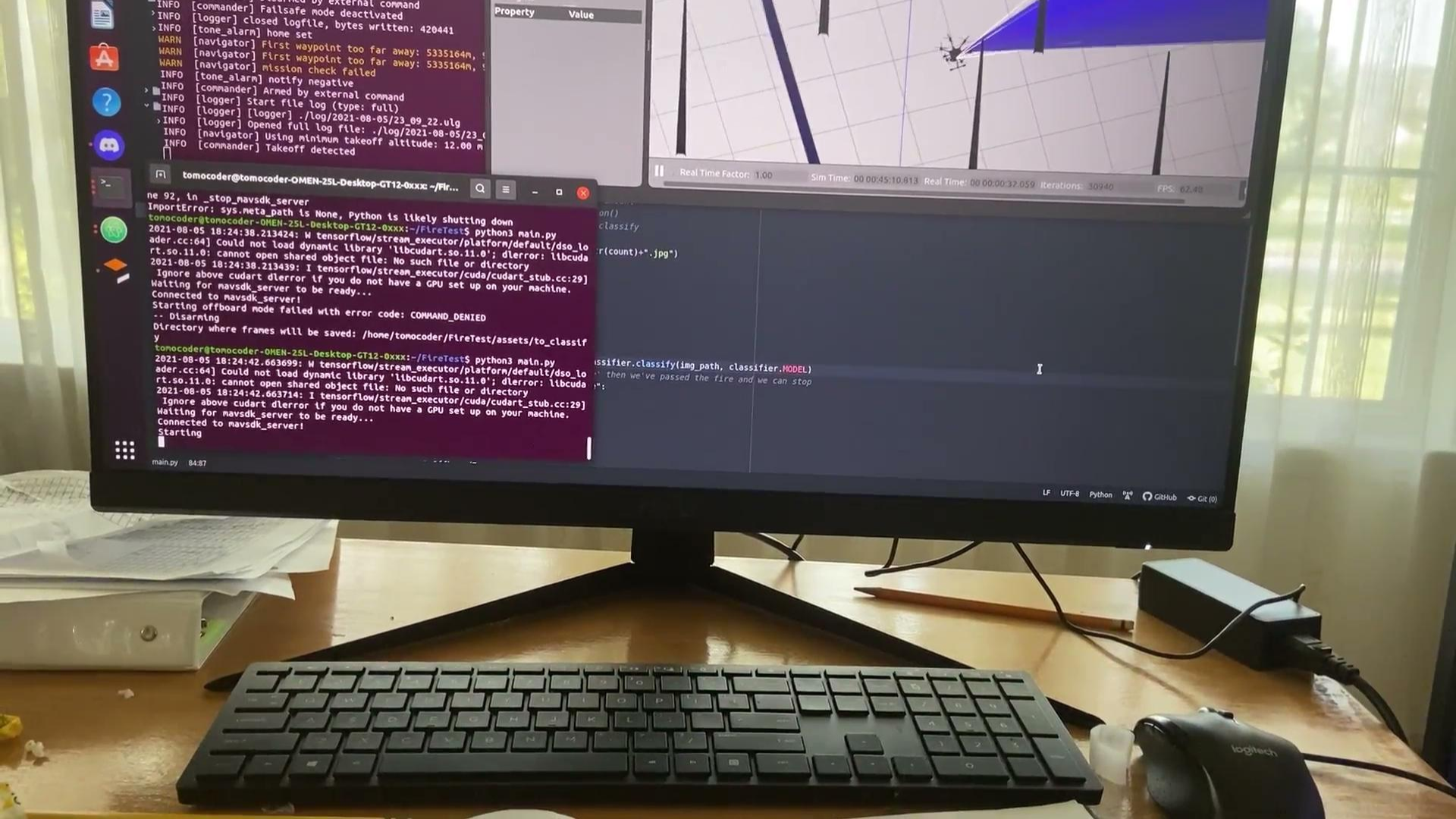
The Drone Script

- MavSdk was used as the architecture.
    - Understanding the async features of MavSdk was a bottleneck.
    - Ultimately we used do_orbit to allow the drone to move in a circle.
    - Await Keyword

```
async for state in drone.core.connection_state():
    if state.is_connected:
        print("Drone discovered")
        break
    await drone.action.arm()
    await drone.offboard.set_position_ned(PositionNedYaw(0.0, 0.0, 0.0, 0.0))
    try:
        await drone.offboard.start()
        print("Starting")
        await drone.action.set_takeoff_altitude(15)
        await drone.action.takeoff()
        # for i in range(120):
            # turn 3 degrees (approx)

        await asyncio.sleep(10)
        # gps = [Position().latitude_deg, Position().longitude_deg,  Position().absolute_altitu
        # print(gps)
        behavior = OrbitYawBehavior(3)
        await drone.action.do_orbit(0.1,1, behavior, 0,0,15)
        # save camera feed to assets/to_classify
        # classify camera feed
```

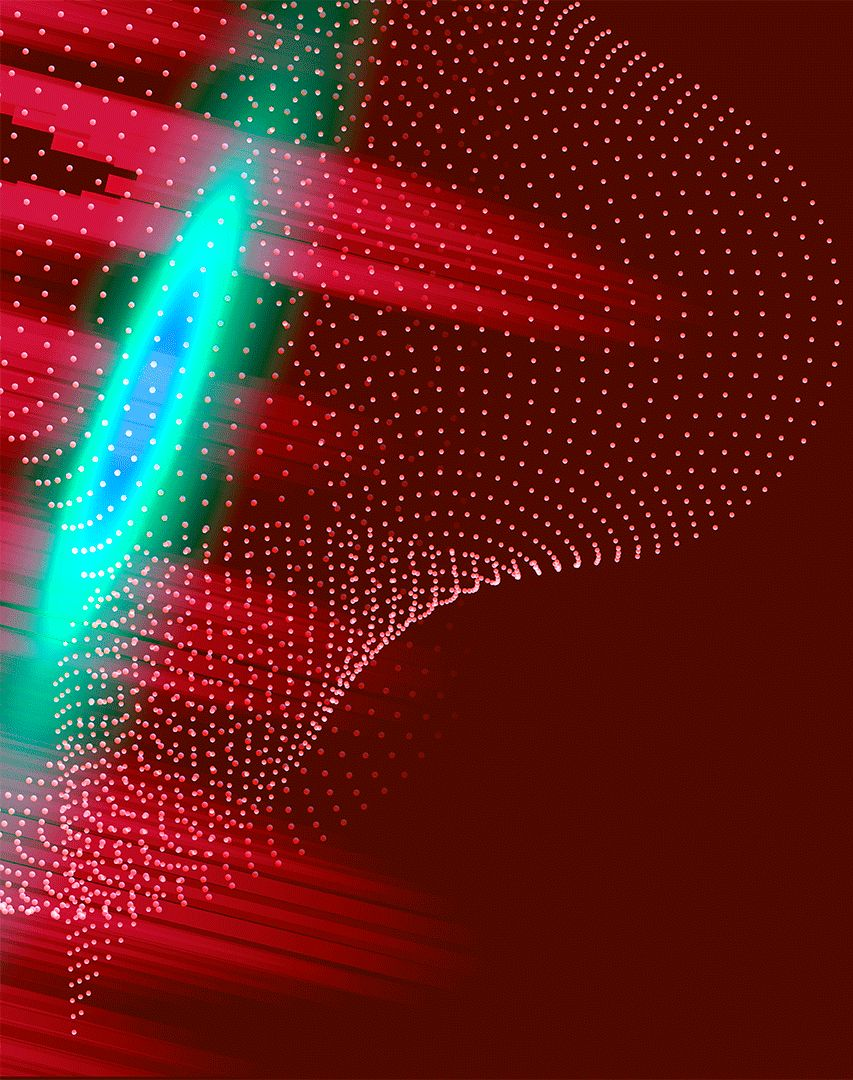
Camera View (POV of Drone)




INFO [commander] Failsafe mode deactivated
INFO [logger] closed logfile, bytes written: 420441
INFO [tone_alarm] home set
WARN [navigator] First waypoint too far away: 5335164m,
WARN [navigator] First waypoint too far away: 5335164m,
WARN [navigator] mission check failed
INFO [tone_alarm] notify negative
INFO [commander] Armed by external command
INFO [logger] Start file log (type: full)
INFO [logger] [logger] ./log/2021-08-05/23_09_22.ulg
INFO [navigator] Opened full log file: ./log/2021-08-05/23_
INFO [commander] Takeoff detected
Property
Value
Real Time Factor: 1.00
Sim Time: 00 00:45:10.813
Real Time: 00 00:00:32.059
Iterations: 30940
FPS: 67.48
tomocoder@tomocoder-OMEN-25L-Desktop-GT12-0xxx: ~/Fir...
ne 92, in _stop_mavsdk_server
ImportError: sys.meta_path is None, Python is likely shutting down
tomocoder@tomocoder-OMEN-25L-Desktop-GT12-0xxx:~/FireTest$ python3 main.py
2021-08-05 18:24:38.213424: W tensorflow/stream_executor/platform/default/dso_lo
ader.cc:64] Could not load dynamic library 'libcudart.so.11.0'; dlerror: libcuda
rt.so.11.0: cannot open shared object file: No such file or directory
2021-08-05 18:24:38.213439: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlerror if you do not have a GPU set up on your machine.
Waiting for mavsdk_server to be ready...
Connected to mavsdk_server!
Starting offboard mode failed with error code: COMMAND_DENIED
-- Disarming
Directory where frames will be saved: /home/tomocoder/FireTest/assets/to_classif
y
tomocoder@tomocoder-OMEN-25L-Desktop-GT12-0xxx:~/FireTest$ python3 main.py
2021-08-05 18:24:42.663699: W tensorflow/stream_executor/platform/default/dso_lo
ader.cc:64] Could not load dynamic library 'libcudart.so.11.0'; dlerror: libcuda
rt.so.11.0: cannot open shared object file: No such file or directory
2021-08-05 18:24:42.663714: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlerror if you do not have a GPU set up on your machine.
Waiting for mavsdk_server to be ready...
Connected to mavsdk_server!
Starting
main.py 84:87
LF UTF-8 Python GitHub Git (0)
logitech

Issues Regarding our Drone (and how they were Overcome)

- There was no ability to work the camera with the original drone (PX4 - Iris).
    - We worked around this issue by changing the model of drone we used. We switched to PX4 - Typhoon H480. This switch allowed us to work with the camera with more freedom.
- Accessing camera information:
    - A video class from Github using OpenCv was able to obtain each frame from the camera. The camera information was ported using a server.
- Implementing neural network alongside async processes.

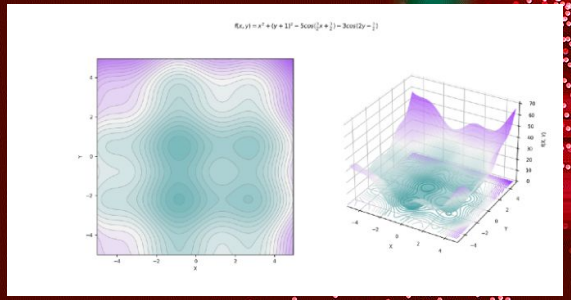


SWARM

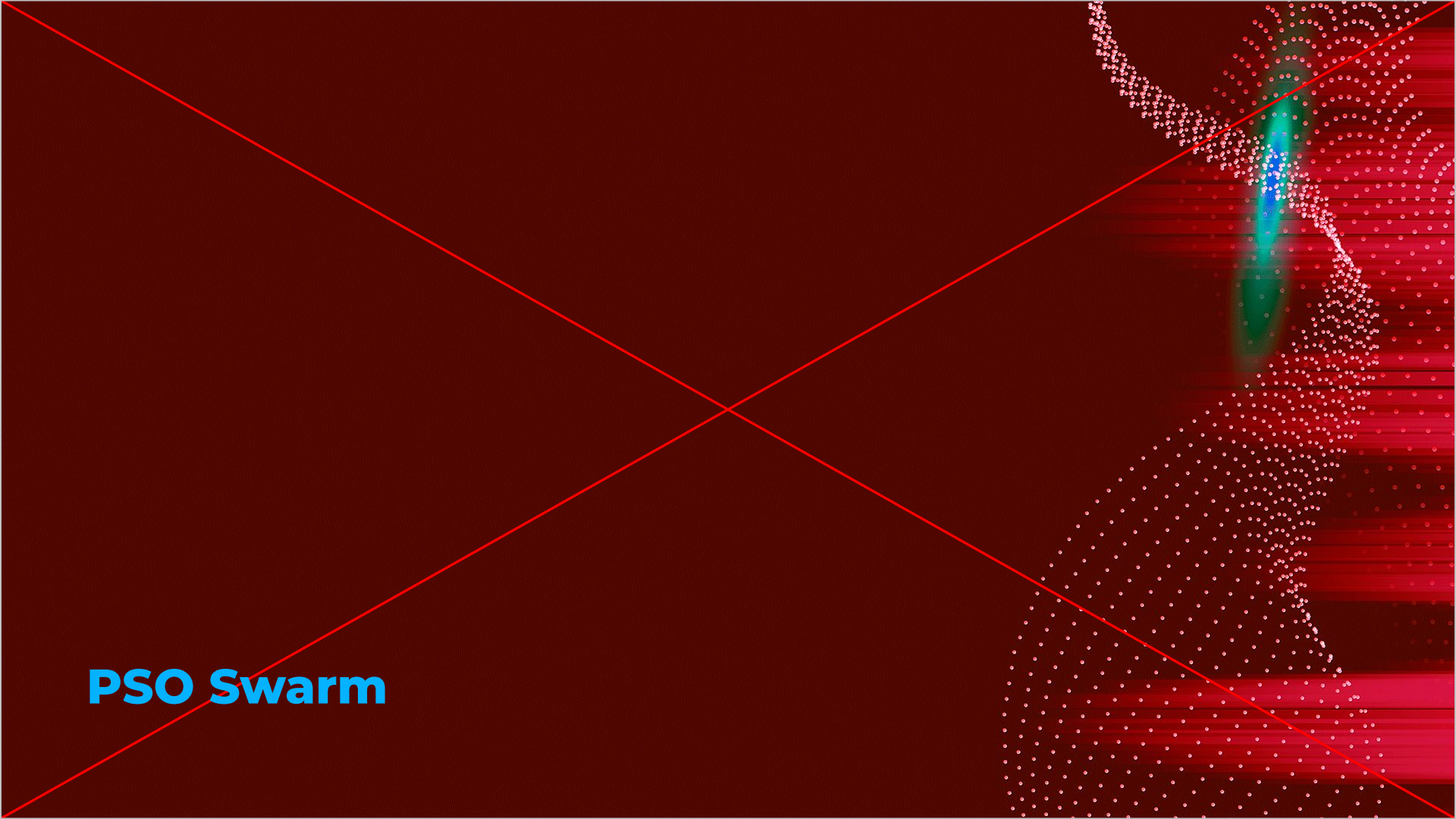
PSO

What is PSO and Why Swarms?

- Swarms increase efficiency and robustness regarding searching.
- Decentralized algorithms are more resilient as if one node (drone) goes down, the rest still remain.
- PSO uses behavior of a single drone and the whole swarm to determine movement to ultimately reach global minimum.

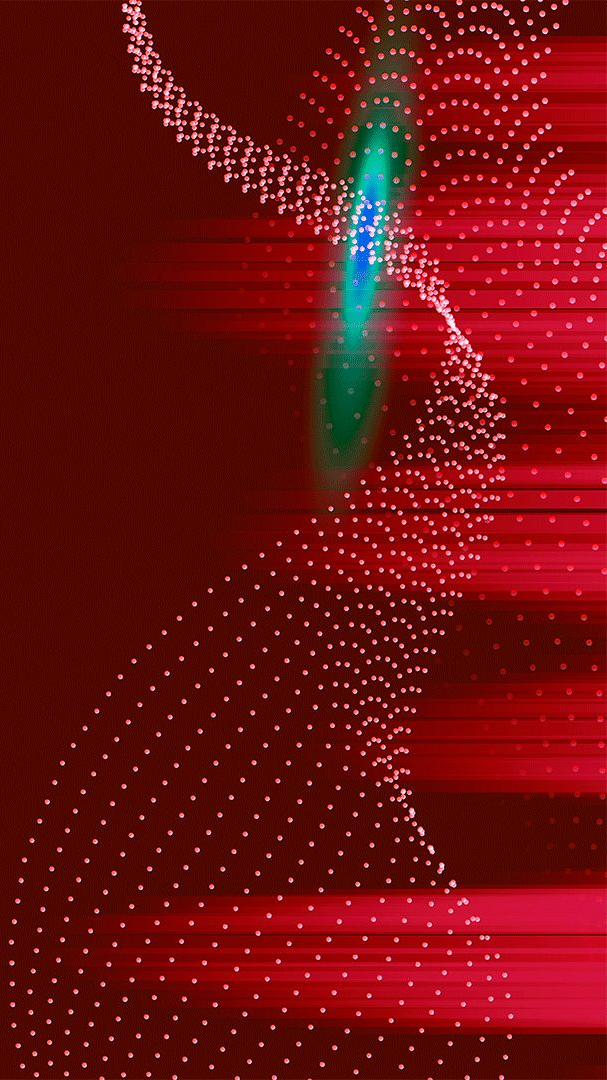


```
void moveParticles(){

    var newVelocities = new List<Vector2>();
    foreach(Vector2 v in velocities){
        newVelocities.Add(v*w);

    }


    var r_1 = new List<Vector2>();
    for(var i = 0;i<N;i++){
        var rand = UnityEngine.Random.Range(0f,1f);
        r_1.Add(new Vector2(rand,rand));
    }
    for(var i = 0; i<newVelocities.Count; i++){


        var vel = new Vector2(c_1*r_1[i].x*(pbests[i].x-particles[i].x), this.c_1*r_1[i].y*(this.pbests[i].y-this.particles[i].y));

        newVelocities[i] += vel;
        // Debug.Log("First Vel "+newVelocities[i]);

    }

    var r_2 = new List<Vector2>();
    for(var i = 0;i<N;i++){
        var rand = UnityEngine.Random.Range(0f,1f);
        r_2.Add(new Vector2(rand,rand));

    }
    for(var i = 0; i<newVelocities.Count; i++){

        newVelocities[i] += new Vector2(this.c_2*r_2[i].x*(this.gbest[i].x-this.particles[i].x), this.c_2*r_2[i].y*(this.gbest[i].y-this.particles[i].y));

    }
```


PSO Swarm

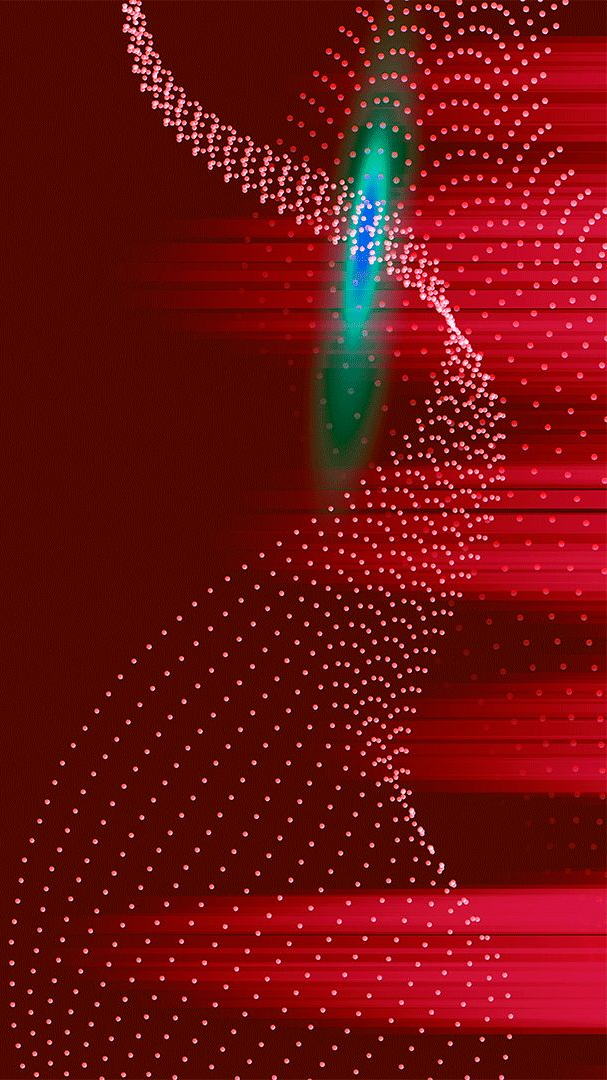
Areas of Improvement

- Improve visual quality/graphics by using another simulator (Ex: AirSim)
- Minimize the amount of delays in simulation
- Expand existing technology to more drones, to get the drones to act and think in a pack.
- Improve pattern drone takes and drone script.
- Use better texturing + models for fire, to boost accuracy of neural network detection of fire.
- Add improved annotations of the frames.



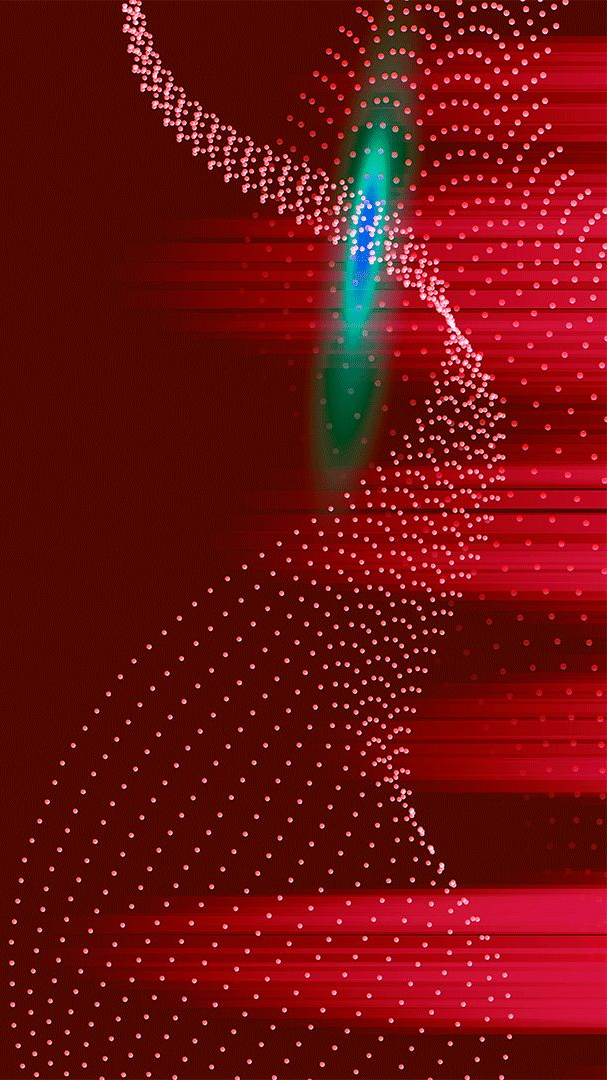
Future Research

- Neural Networks
  - Implementation of Neural Net with Drone (work in progress)
  - **Simulation end**
    - Improve realism - dynamic fire, large forest, randomized fire location



Future Research

- Cohort Intelligence
  - Computationally light
  - Drone swarm control
    - Powerful
    - Fast
    - Sensitive initial parameters
  - Highly scalable for large forest applications
- Atmosphere/temp



Resources

- upGrad. (2020, June 19). *Neural network tutorial: Step-by-step guide for beginners*. upGrad blog. https://www.upgrad.com/blog/neural-network-tutorial-step-by-step-guide-for-beginners/.
- Innocente, M.S.; Grasso, P. Swarm of Autonomous Drones Self-Organised to Fight the Spread of Wildfires. In Proceedings of the GEOSAFE Workshop on Robust Solutions for Fire Fighting (CEUR), L'Aquila, Italy, 19–20 July 2018
- Kulkarni, A. J., Baki, M. F., & Chaouch, B. A. (2016). Application of the cohort-intelligence optimization method to three selected combinatorial optimization problems. European Journal of Operational Research, 250(2), 427–447. doi:10.1016/j.ejor.2015.10.008
- Mac, T. T., Copot, C., Duc, T. T., & De Keyser, R. (2016). AR.Drone UAV control parameters tuning based on particle swarm optimization algorithm. 2016 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR). doi:10.1109/aqtr.2016.7501380
- Ackerman, E. (n.d.). *This Autonomous Quadrotor Swarm Doesn't Need GPS*. IEEE Spectrum: Technology, Engineering, and Science News. https://spectrum.ieee.org/automaton/robotics/drones/this-autonomous-quadrotor-swarm-doesnt-need-gps.

Resources Continued...

- Popoago. (2019, October 11). *Detecting Fires using Tensorflow*. Medium. https://medium.com/analytics-vidhya/detecting-fires-using-tensorflow-b5b148952495.
- Kulkarni, Anand J., et al. “Cohort Intelligence: A Self Supervised Learning Behavior.” *IEEE Xplore*, 1 Oct. 2013, https://ieeexplore.ieee.org/document/6721994 . Accessed 18 July 2021.
- “Improved Cohort Intelligence—a High Capacity, Swift and Secure Approach on JPEG Image Steganography.” *Journal of Information Security and Applications*, vol. 45, 1 Apr. 2019, pp. 90–106, https://www.sciencedirect.com/science/article/abs/pii/S2214212618304824 , 10.1016/j.jisa.2019.01.002. Accessed 18 July 2021.
- “Investigation on Biomedical Waste Management of Hospitals Using Cohort Intelligence Algorithm.” *Soft Computing Letters*, vol. 3, 1 Dec. 2021, p. 100008, https://www.sciencedirect.com/science/article/pii/S2666222120300071 , 10.1016/j.socl.2020.100008. Accessed 18 July 2021.
- Patil, Mukundraj V., and Anand J. Kulkarni. “Pareto Dominance Based Multiobjective Cohort Intelligence Algorithm.” *Information Sciences*, vol. 538, 1 Oct. 2020, pp. 69–118, https://www.sciencedirect.com/science/article/abs/pii/S0020025520304199 , 10.1016/j.ins.2020.05.019. Accessed 18 July 2021.

Resources Continued...

- Mavlink. “Mavlink/MAVSDK-Python: MAVSDK Client for Python.” *GitHub*, github.com/mavlink/MAVSDK-Python.
- Robmarkcole. “Robmarkcole/Fire-Detection-From-Images: Detect Fire in Images Using Neural Nets.” *GitHub*, Aug. 2020, github.com/robmarkcole/fire-detection-from-images.
- “# Gazebo Simulation.” *Gazebo Simulation | PX4 User Guide*, docs.px4.io/master/en/simulation/gazebo.html.
- Garberoglio, Leonardo. “What Is Default Gps Coordinate for px4 Sitl Gazebo? - PX4 Forum.” *Discussion Forum, for PX4, Pixhawk, QGroundControl, MAVSDK, MAVLink*, 1 June 2019, discuss.px4.io/t/what-is-default-gps-coordinate-for-px4-sitl-gazebo/11224.
- “Class Action.” *Class Action · MAVSDK Guide*, mavsdk.mavlink.io/v0.37.0/en/api_reference/classmavsdk_1_1_action.html.
- “# Gazebo Simulation.” *Gazebo Simulation | PX4 User Guide*, docs.px4.io/master/en/simulation/gazebo.html.
- Raul Alvarez-Torrico : raul@tecbolivia.com
- -For providing documentation for Gazebo PX4 install
- Torricocircuitcellar.com/author/raul-alvarez-torrico/, Raul Alvarez. “Writing MAVSDK/PX4 Drone Applications.” *Circuit Cellar*, 1 Aug. 2020, circuitcellar.com/research-design-hub/design-solutions/writing-mavsdk-px4-drone-applications/.

Thank you, any Questions?