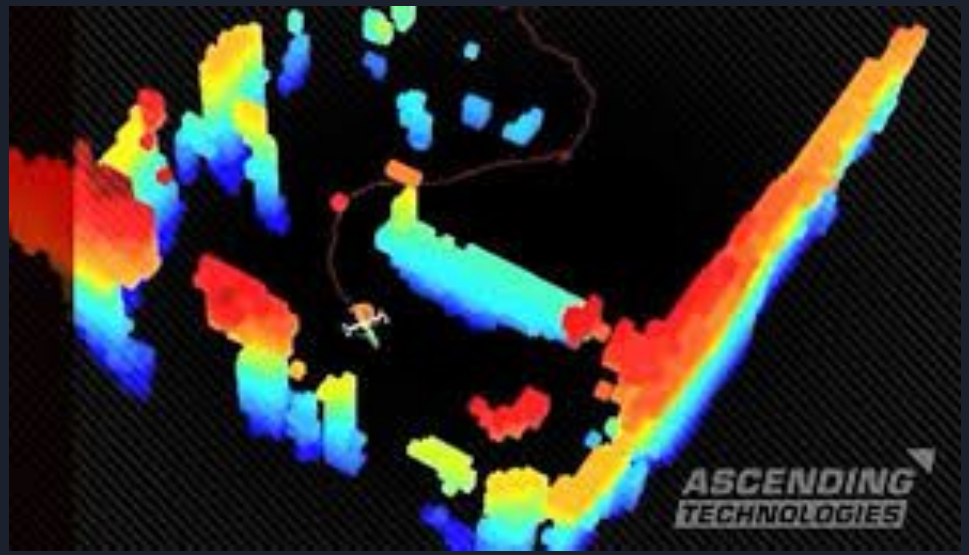
Auto**Quads Final Presentation**

Spring 2020 Senior Design

Max Rudolph
Eddie Stevens
Dave Patel
Rishov Sarkar
Michael Bermudez
Nyaire Najieb
Suhani Jain

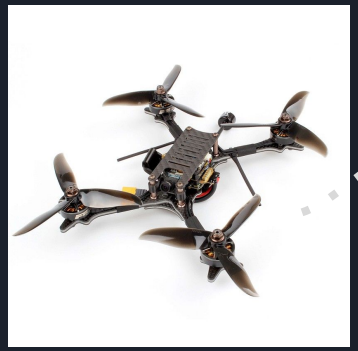
Our Original Goal

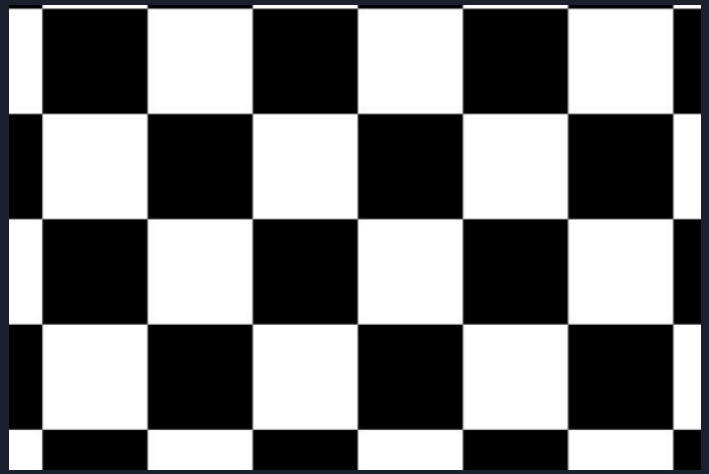
- Build a drone that can autonomously fly through a racing course (with course knowledge apriori)



Our Revised Goal (Post COVID-19)

- Have a drone “fly” autonomously using real-time computer vision and tracking
- Simulate all the necessary components to deliver a final product
- Prepare necessary software and hardware for next senior design team to start the the project right off the bat (without being delayed, like we were)

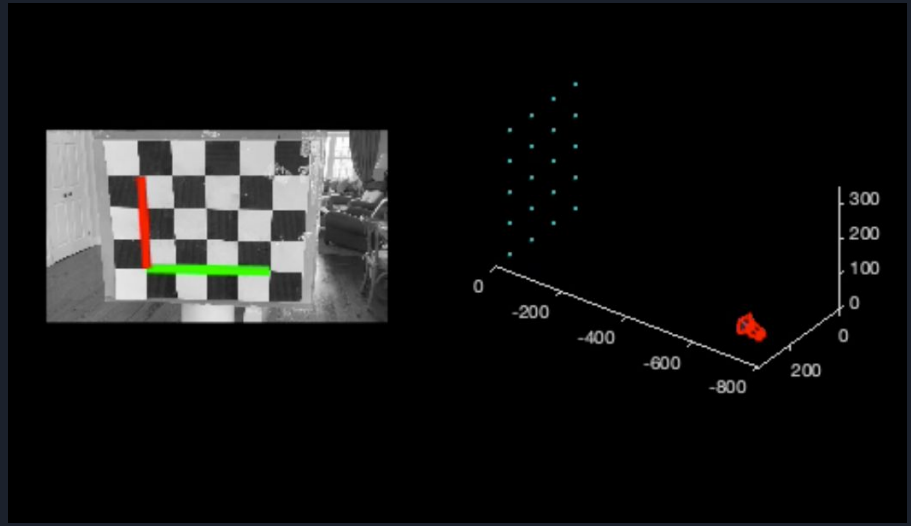




What we accomplished

- Developed real-time computer vision localization scripts using MATLAB and OpenCV
- Used localization to create velocity commands to direct drone to target
- Implemented drone simulator that mimics the setup procedure of actual ArduPilot on drone
- Built actual drone that takes off autonomously and loiters for commands

```
(drone) Maxs-MacBook-Pro-3:velocity_control maxrudolph$ python ve
Start simulator (SITL)
Starting copter simulator (SITL)
SITL already Downloaded and Extracted.
Ready to boot.
Connecting to vehicle on: tcp:127.0.0.1:5760
CRITICAL:autopilot:APM:Copter V3.3 (d6053245)
CRITICAL:autopilot:Frame: QUAD
CRITICAL:autopilot:Calibrating barometer
CRITICAL:autopilot:Initialising APM...
CRITICAL:autopilot:barometer calibration complete
CRITICAL:autopilot:GROUND START
Basic pre-arm checks
 Waiting for vehicle to initialise...
 Waiting for vehicle to initialise...
 Waiting for vehicle to initialise...
 Waiting for vehicle to initialise...
```



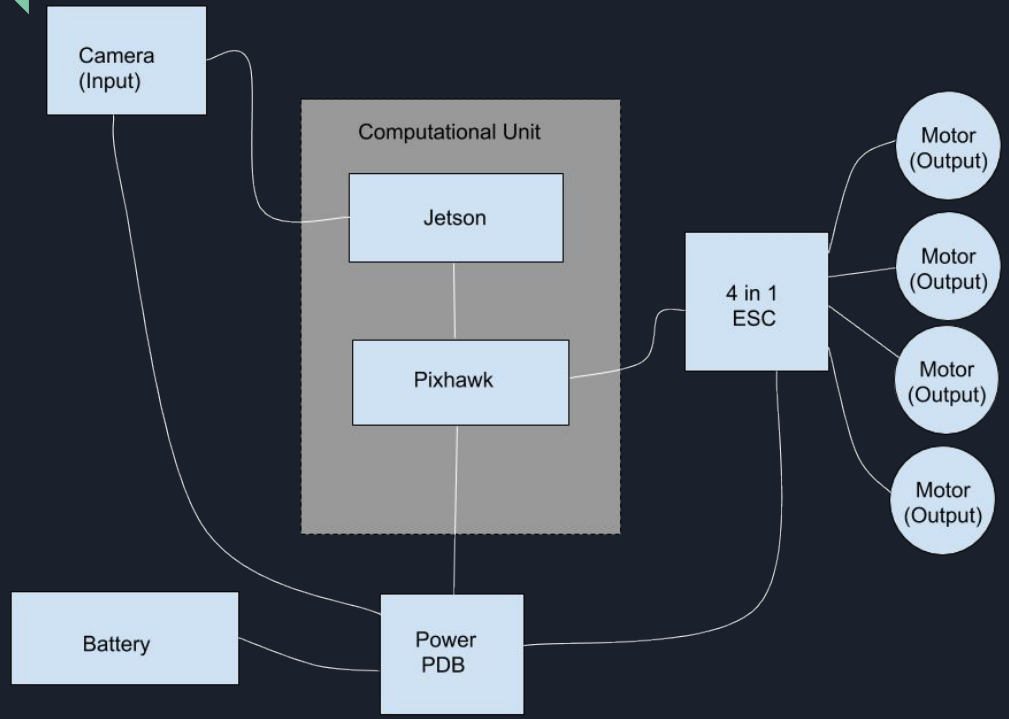
How have we measured success (post-COVID)?

- Comparing CV computed path with dronekit simulator path
  - Our simulator can do take in position commands (absolute and relative), loiter commands, and velocity commands.
  - We relegated ourselves to just sending velocity commands to mimic real life drone control
  - Succeeded

- Achieving possible real-time CV for drone implementation
  - Autonomous drone control is only possible in real-time
  - Reduce computation time for traditional CV methods which will allow for real time control of drone once the next team integrates the software
  - Succeeded

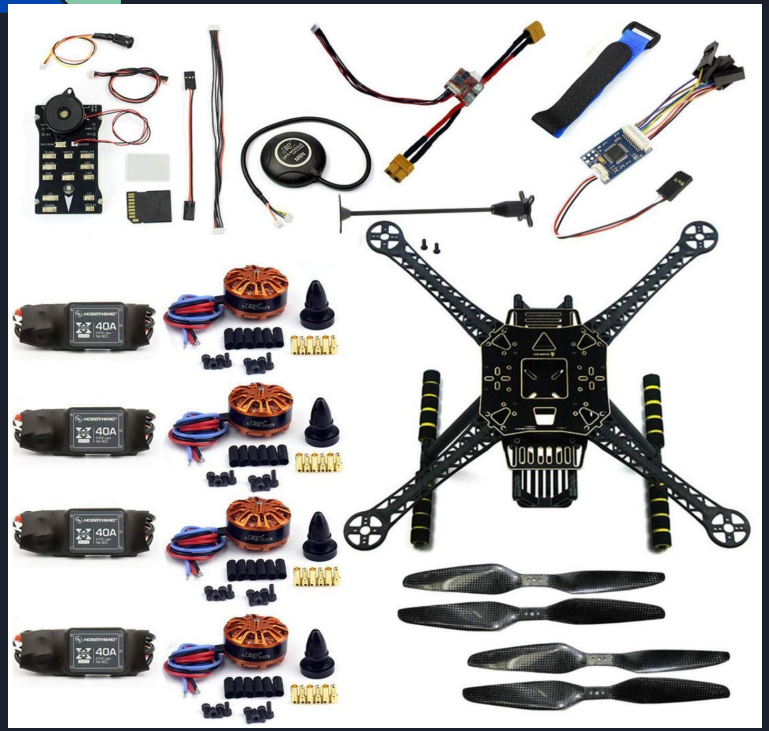
Drone Architecture and Software Stack

Proposed Flight System



- **Nvidia Jetson Nano**: computer used to perform computer vision and path planning.
- **Pixhawk**: flight controller to keep drone in air.
- **ESC:** Electronic Speed Controller used to send signals directly to motor.

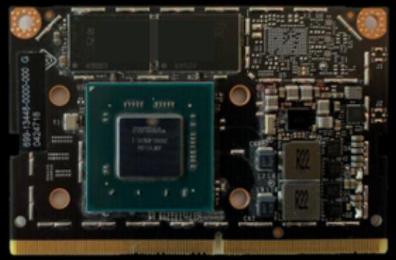
Hardware: Drone Kit



- Most work on drone system will be in software, not hardware; perception, control, data networking, path planning
- Drone kits are cheap, readily available, and reliable.
- Individual parts can be swapped out if they break cheaply and with minimal effort

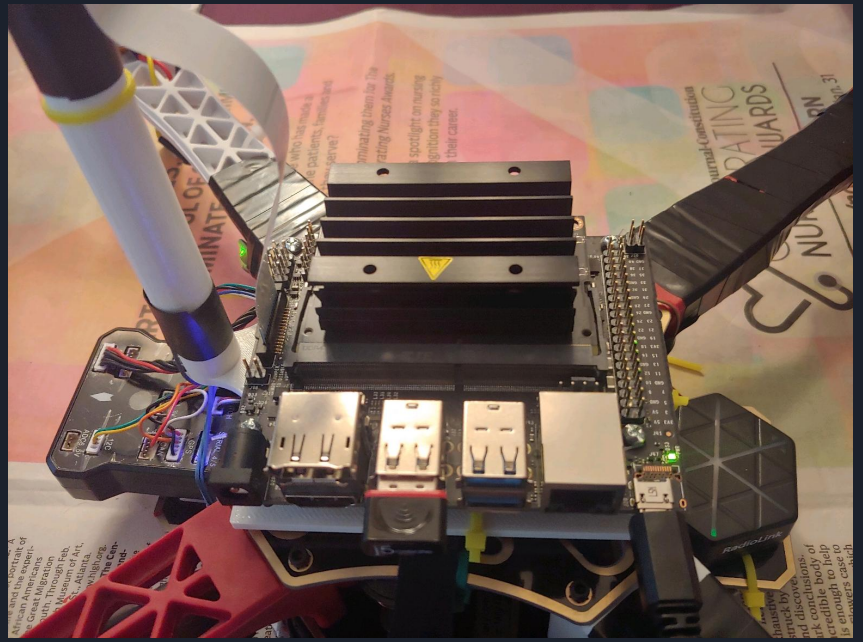
Computing Unit

JETSON NANO SPECIFICATIONS



GPU	128 Core Maxwell 472 GFLOPs (FP16)
CPU	4 core ARM A57 @ 1.43 GHz
Memory	4 GB 64 bit LPDDR4 25.6 GB/s
Storage	16 GB eMMC
Video Encode	4K @ 30 | 4x 1080p @ 30 | 8x 720p @ 30 (H.264/H.265)
Video Decode	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 16x 720p @ 30 | (H.264/H.265)
Camera	12 (3x4 or 4x2) MIPI CSI-2 DPHY 1.1 lanes (1.5 Gbps)
Display	HDMI 2.0 or DP1.2 | eDP 1.4 | DSI (1 x2) 2 simultaneous
UPHY	1 x1/2/4 PCIE 1 USB 3.0
SDIO/SPI/SysIOs/GPIOs/I2C	1x SDIO / 2x SPI / 5x SysIO / 13x GPIOs / 6x I2C

Final Build



Estimated Cost

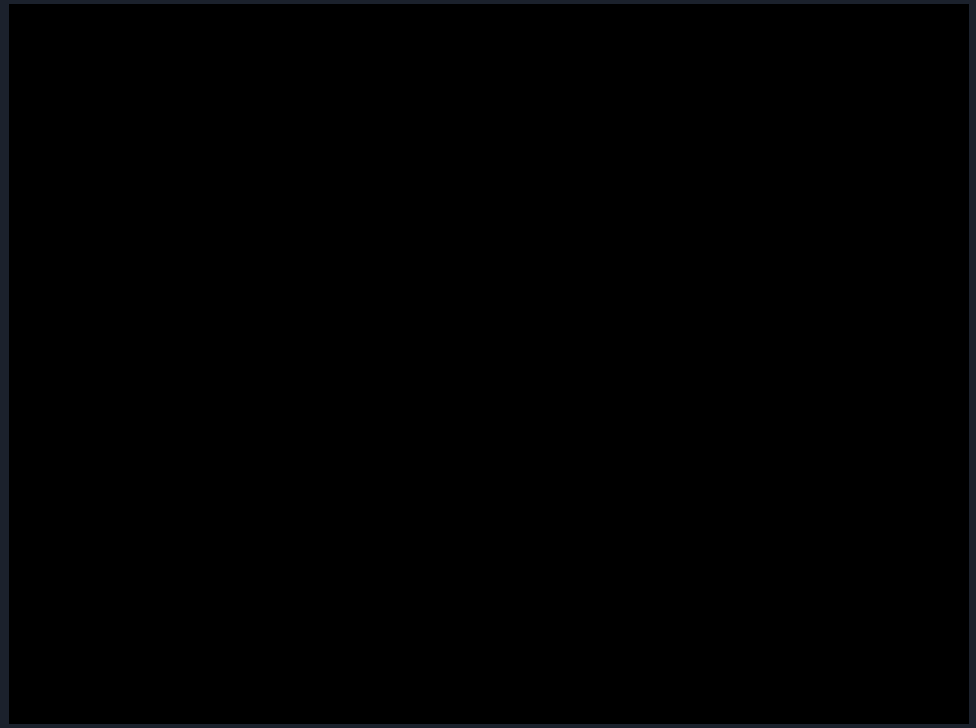
Product Description	Quantity	Unit Price ($)	Total Price ($)
Drone	2	280.00	560.00
NVIDIA Jetson	3	119.00	357.00
Camera	2	80.00	160.00
Batteries	4	20.00	80.00
Obstacles	15	5.00	75.00
Total Cost			1232.00

Capability Methodology and Demonstration

Methodology

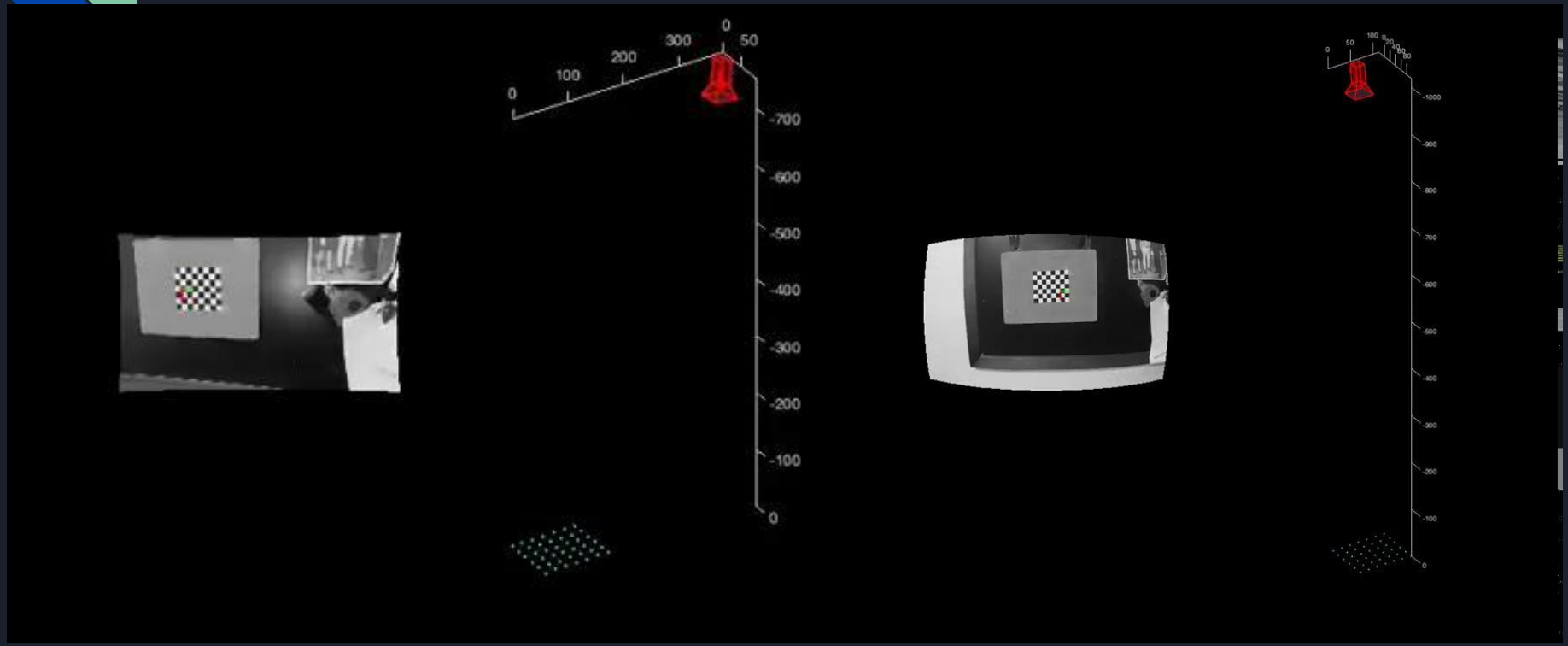
1. Capture simulated video
2. Detect checkerboard of variable size in each video frame
    a. In simulation this step is done at 60Hz but only needs to operate at max. frequency of 10Hz when implemented on the drone because flight controller cannot respond faster than 10Hz
3. Use MATLAB/Python to extract location/orientation data
    a. Plot the calculated location for accuracy measures of drone simulator
4. From any given location, calculate and record velocity necessary to bring drone to target
5. Send real-time calculated velocities to drone simulator
6. Compare dronekit path to MATLAB path to ensure accuracy

Capture Real-Time Localization with Checkerboard

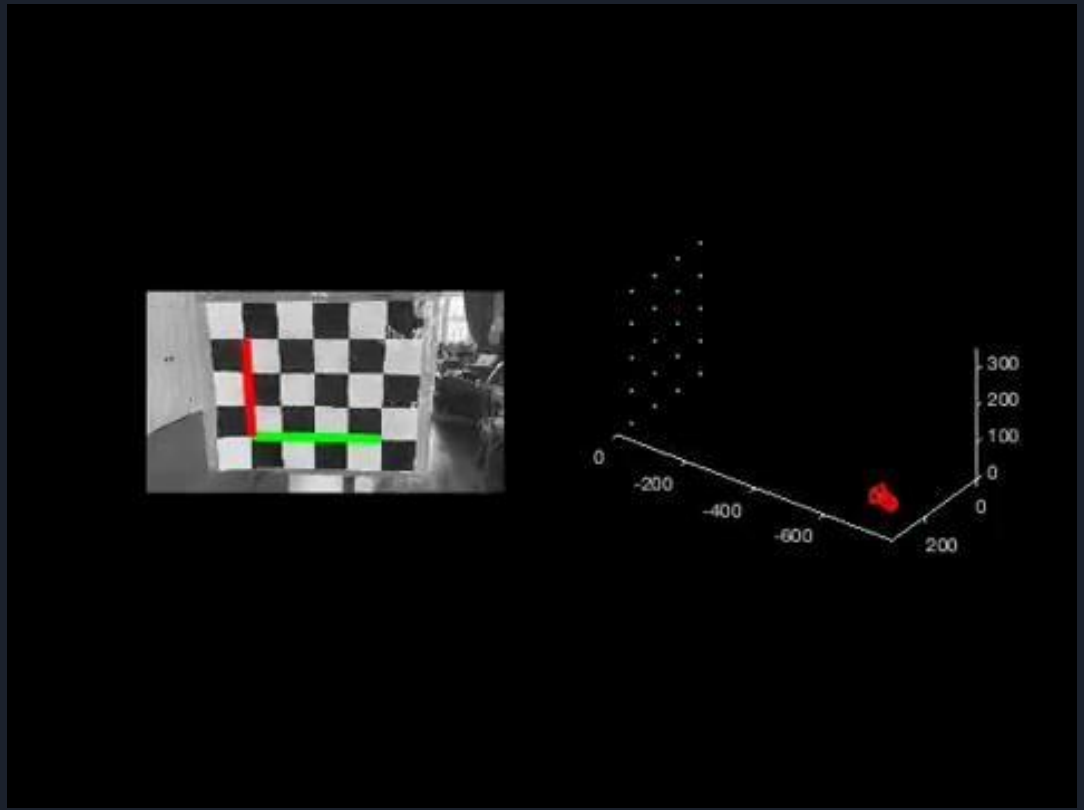


The localization present in the video was calculated in real-time at around 10 Hz.

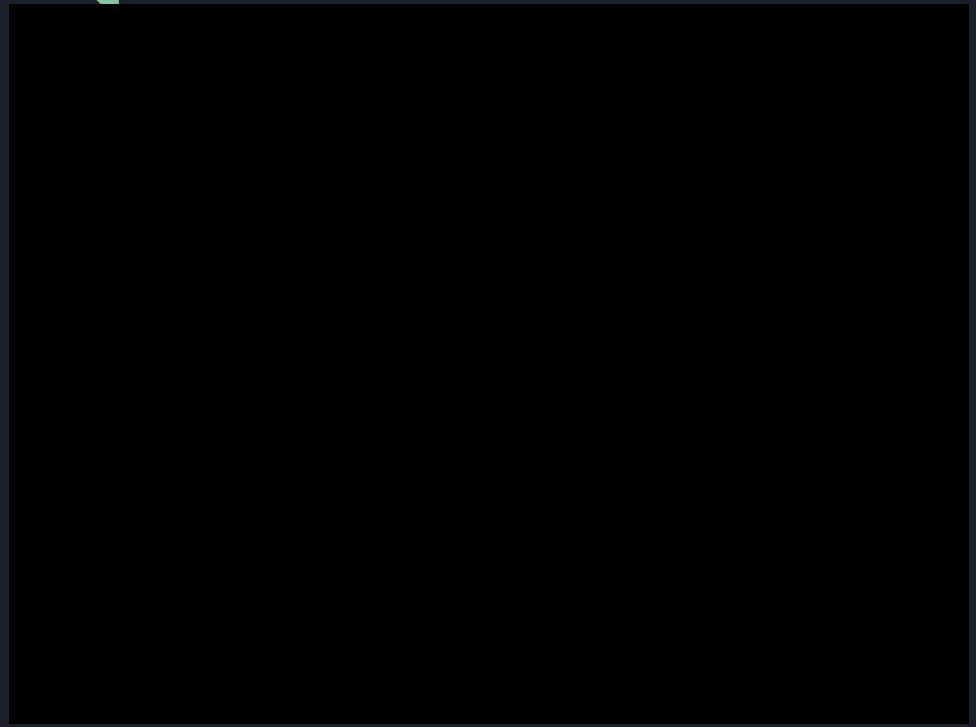
Versatility of Checkerboard Detection



Simulated Drone Movement



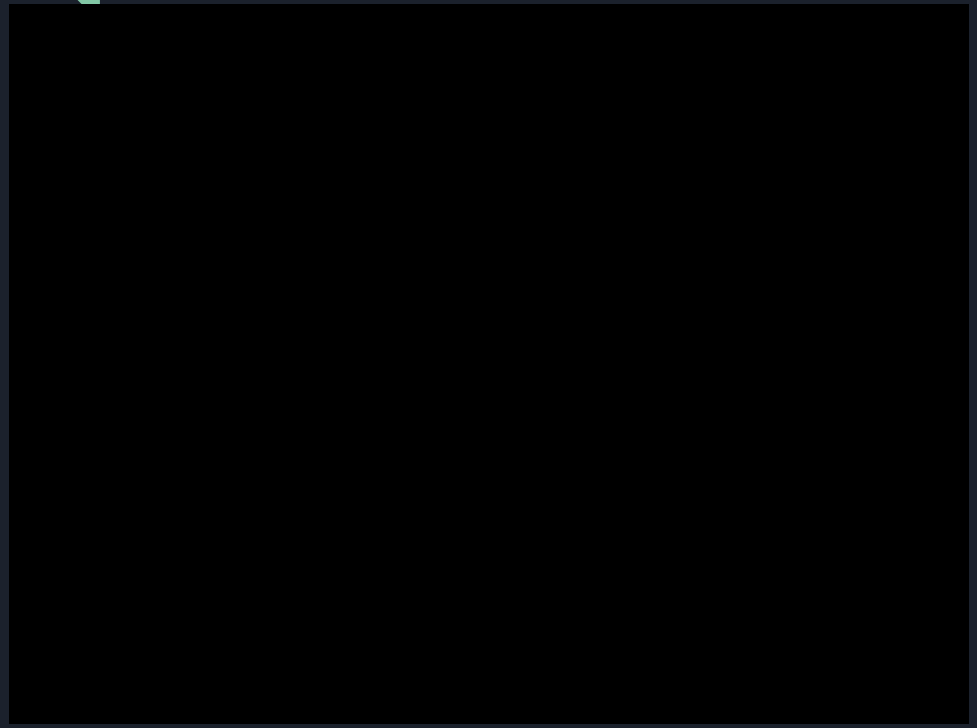
Computed Drone Movement



Pink dots represent points at which drone velocities are calculated and sent to drone simulator (approx. 1 Hz)

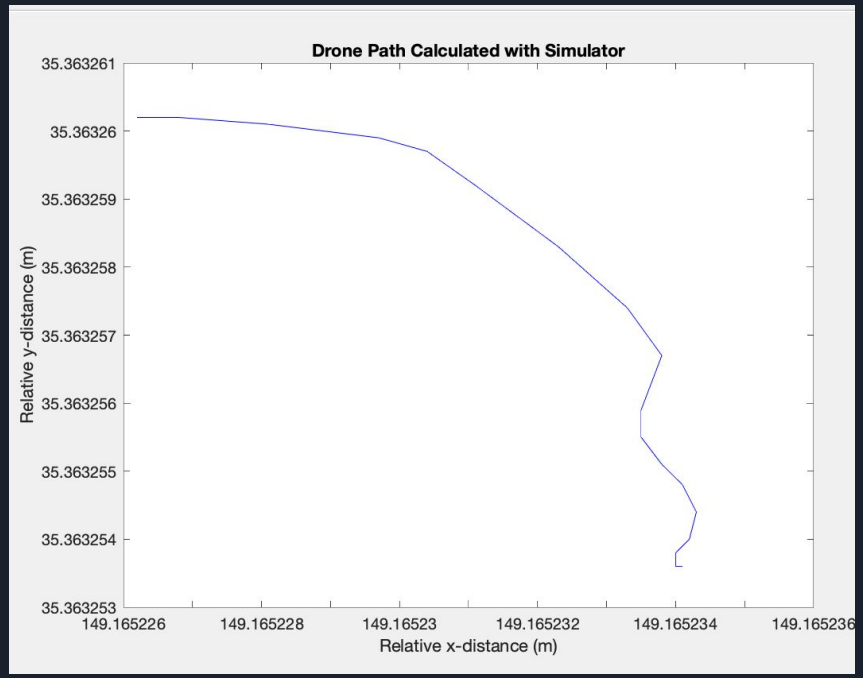
Simulation Examples

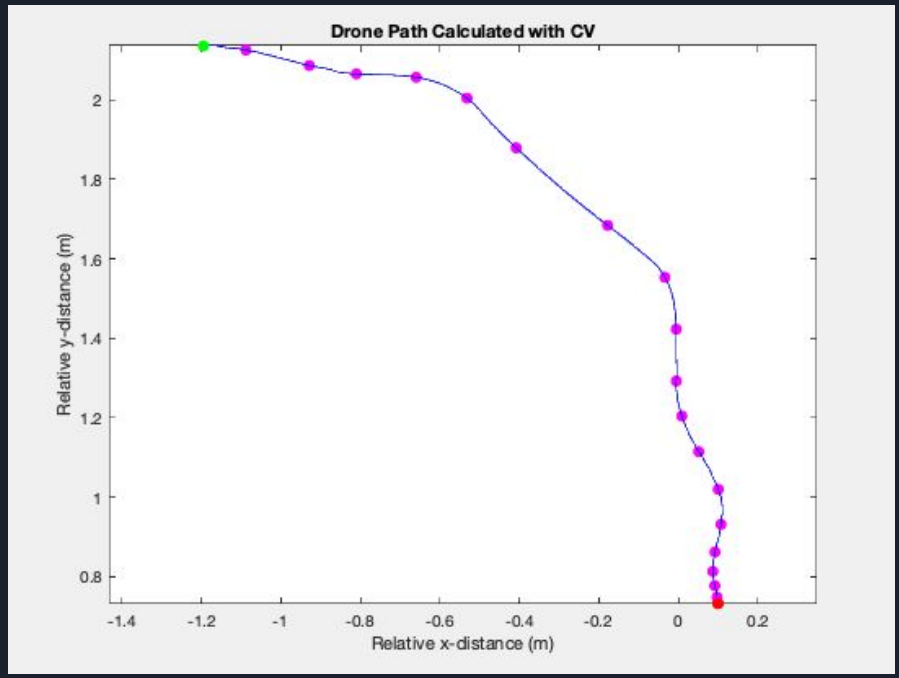
Path of Simulated Drone Using Velocities from CV



Visualization of drone movement when velocities are put into simulator. Notice the matching between this plot and the previous plot

Side by Side Comparison of Drone Paths



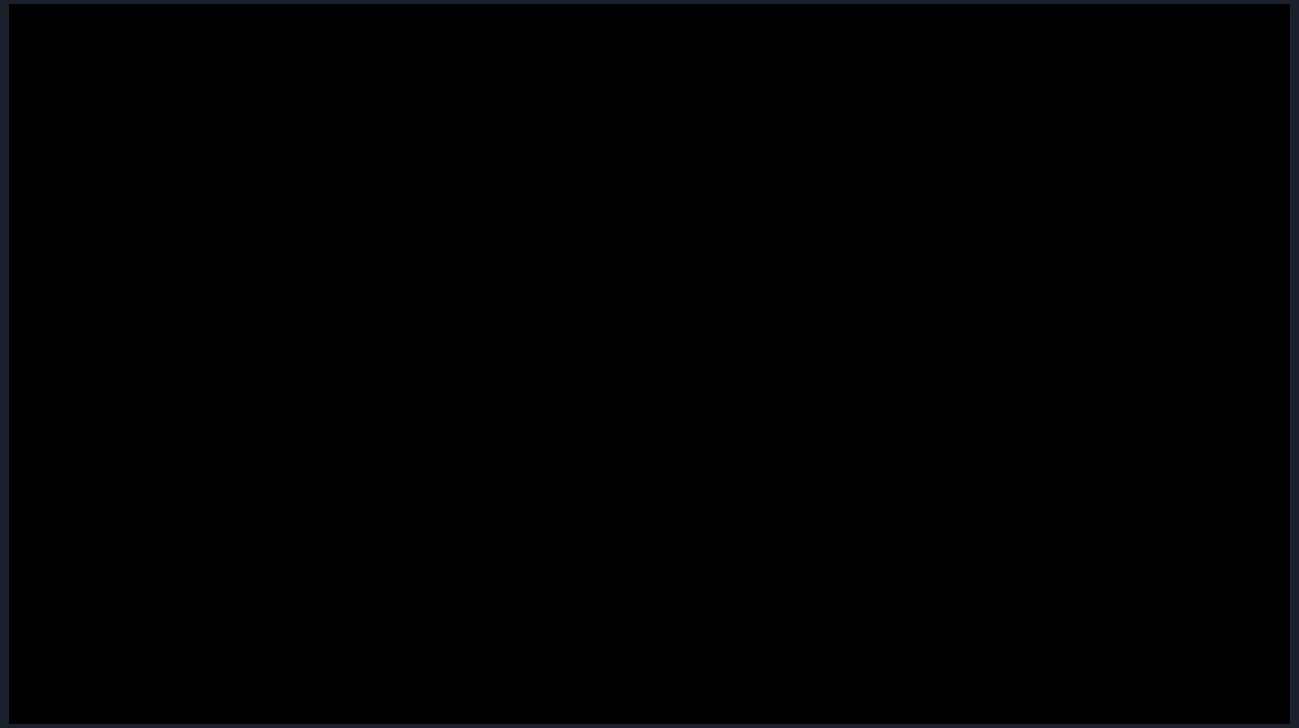


Autonomy on Drone

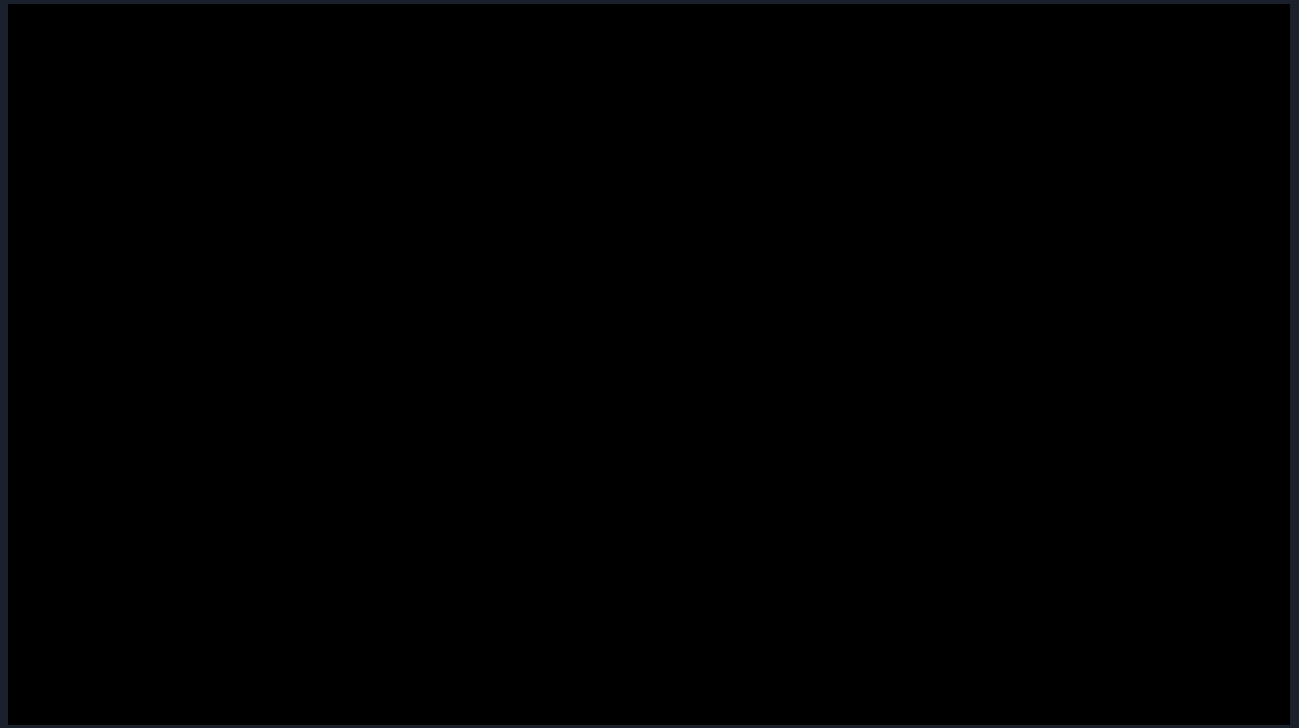
First Steps: Unstable Drone

Self-Stabilization: Loiter Mode

Autonomous Flight in Guided Mode



Footage from Drone



Potential Final Product

- Given our separate demonstrations for object detection and issuing flight commands, we have laid the groundwork for an autonomous drone.

- Future teams could utilize our real-time localization scripts to generate appropriate velocity commands, then use the MAVLink protocol to send the commands to the flight controller. We proved the concepts; they just need to be fully integrated on the drone!

Reflection and Mistakes

- Project scope too large
  - Honing in on a single aspect such as CV and path prediction would have been better
- Having a single Jetson cause delays
  - A second jetson would have allowed for prototype/testing CV algorithms without having drone on hand
- Too much time on drone hardware, not enough time on CV
  - A well established detection and path planning algorithm would have been of greater use to future groups. Our understanding of the drone hardware-software only helped us at the beginning.
  - Lack of clarity on amount of software and hardware integration needed

Future Team Expectations

- Integrate drone localization into actual drone
- Add more autonomous functionality: path planning and execution
- Add computer vision software to enable the camera do detect and recognize objects
- Ability to prevent collisions and re-adjust course with object detection
- After accurate detection is achieved, strive for higher flight velocities
- Attempt a drone race course once the software is fully integrated and optimized
- Multiple or better cameras could be added, for better field of view. This would allow for better forward navigation and obstacle avoidance.

Team Members

- Max Rudolph
- Eddie Stevens
- Dave Patel
- Rishov Sarkar
- Michael Bermudez
- Nyaire Najieb
- Suhani Jain

AutoQuads Team 33

Advisor: Dr. Jennifer Hasler