

# Design and Implementation of an Object-tracking UAV

Developed by: Ryan Lewis

# Need Statement:

Integrate an open source computer vision package into a hobby drone to serve as a platform for development of more advanced object tracking software and other computer vision based augmented flight modes.



# Literature Review:

## Ignacio Mellado's LinkQuad:

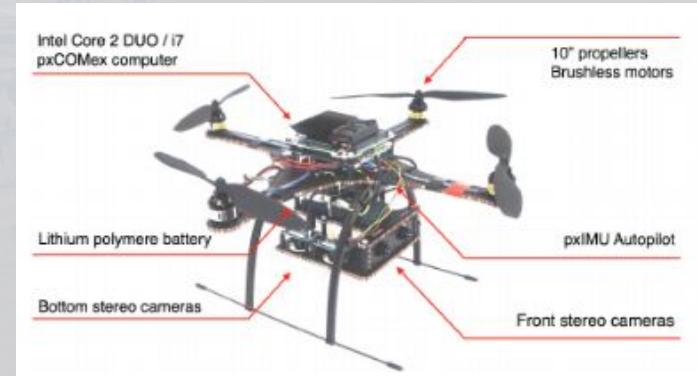
- Autonomously Navigates Using Optical Flow
- 80x60 camera image limits use cases
- <https://vimeo.com/61558982>



# Literature Review:

## The Pixhawk Open-source Computer Vision Framework for MAVs:

- Does not perform object detection in real-time
- Uses stereo cameras for range finding application
- Code unavailable



# Literature Review:

## Skydio Drone:

- Performs object tracking/following locally in real-time
- Commercially available (MSRP \$1000)
- Computer vision only available using limited API



# Literature Review:

## Ryan Lewis' Object Tracking UAV:

- Can be made for less than \$1000
- Utilizes open source hardware, firmware and software
- Tracks objects



# Approach:

Design a system that is accessible to hobby drone builders and product developers

→ **Hardware**

Designs publicly available

→ **Software**

Open source

→ **Availability**

Commercially available components

→ **Safety**

Designed in compliance with *FAA H.R. 302 Sect. 345 - Small unmanned aircraft safety standards*

# Engineering Requirements:

Design a system that is able to:

- Thrust : Weight Ratio  
Greater than 1.1
- Perform Local Vision Processing  
Does not require video stream to remote computer for image processing
- Track Green Objects  
Ranging from (29, 86, 6) to (64, 255, 255) in the HSV colorspace

# Constraints and Standards:

The UAV and object tracking system were implemented and tested in accordance with the following standards:

- IEEE 802.11  
Wireless LAN communication standards
- FCC Title 47 C.F.R. § 2.106  
Frequency allocations: used for telemetry comms
- Public Law 112-95 Section 336  
Operating requirements for non-commercial drone use

# Impacts:

## → Welfare

Can be used as a platform to track and disable insurgents in countries containing resources which are essential to the continued operation of the United States military industrial complex.

## → Public Health

Using micro-aerial-vehicles to protect assets which are vital for the operation of the United States military industrial complex could aid in subduing insurgent efforts without causing excessive collateral damage

# Impacts:

## → Safety

Using remotely operated systems to track points of interest and key targets could allow US soldiers to make more informed decisions when advancing into hostile territories.

## → Cultural

The use of this platform could allow the US to remain anonymous while gathering critical tactical information and allow for plausible deniability in scenarios which incur non-optimal collateral damage.

# Impacts:

## → Global

Implementing additional Micro Aerial Vehicles in intelligence gathering would accelerate the global arms race and subsequently stimulate the global economy as counter measures are developed by foreign nations

## → Social

Being able to gather information and potentially disarm insurgents in foreign nations anonymously could allow for better US/Foreign relations

# Impacts:

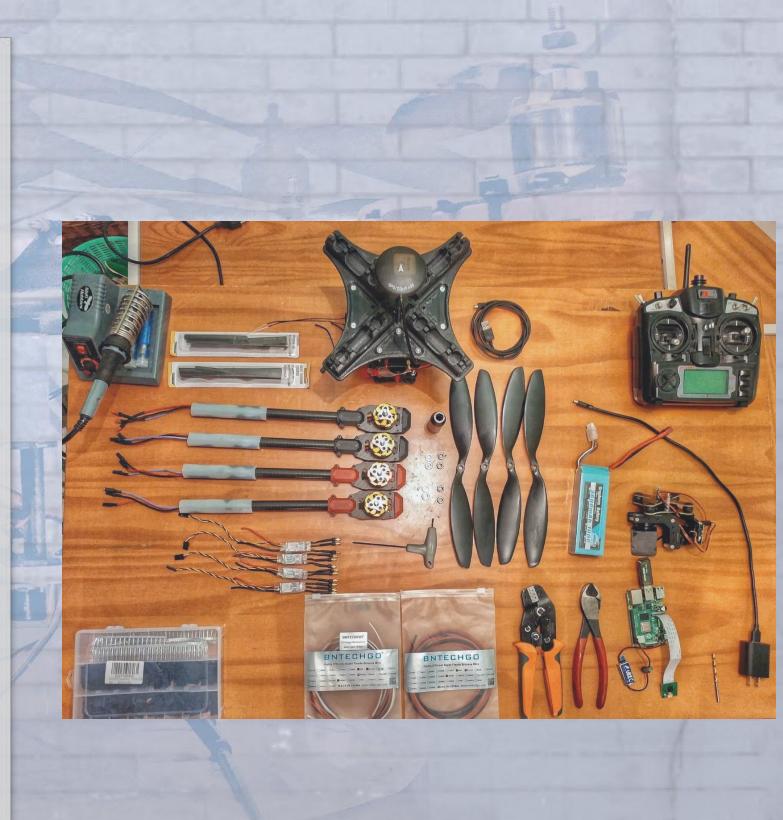
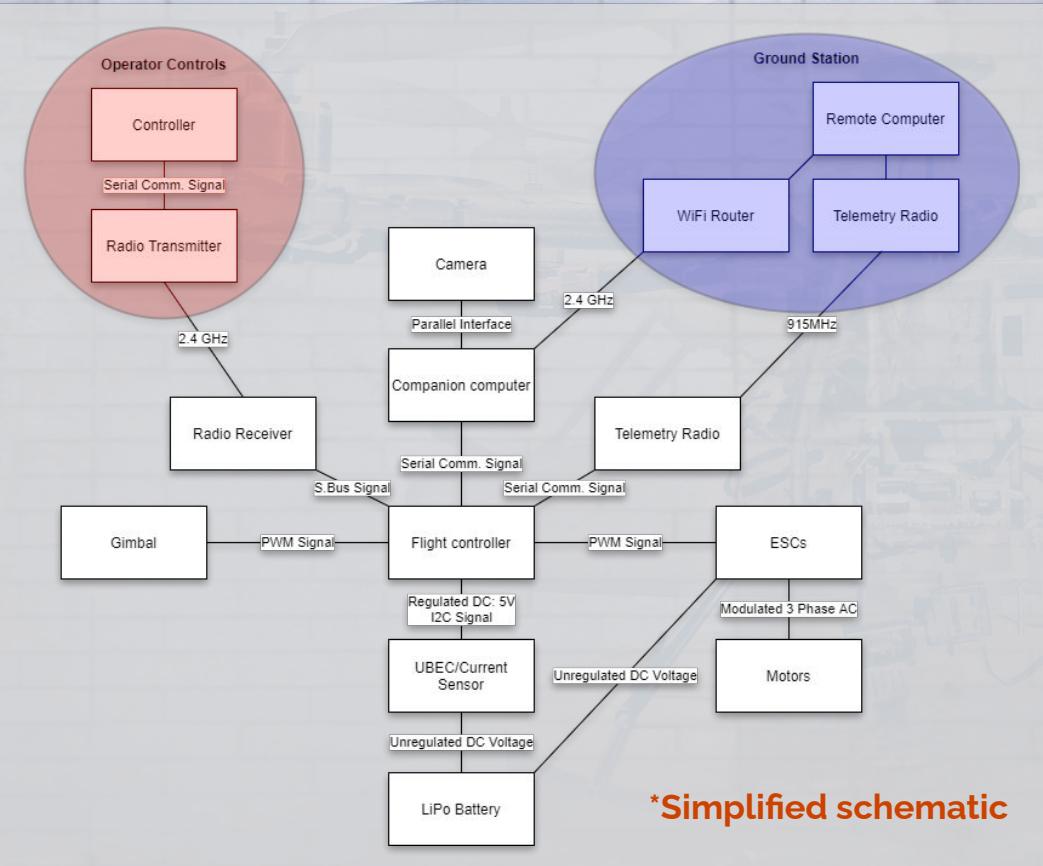
## → Environmental

The use of rechargeable lithium batteries allows for zero on-site emissions while financially supporting artisanal mineral mines which use local labor to gather cobalt (as opposed to industrial operations which produce higher levels of pollution)

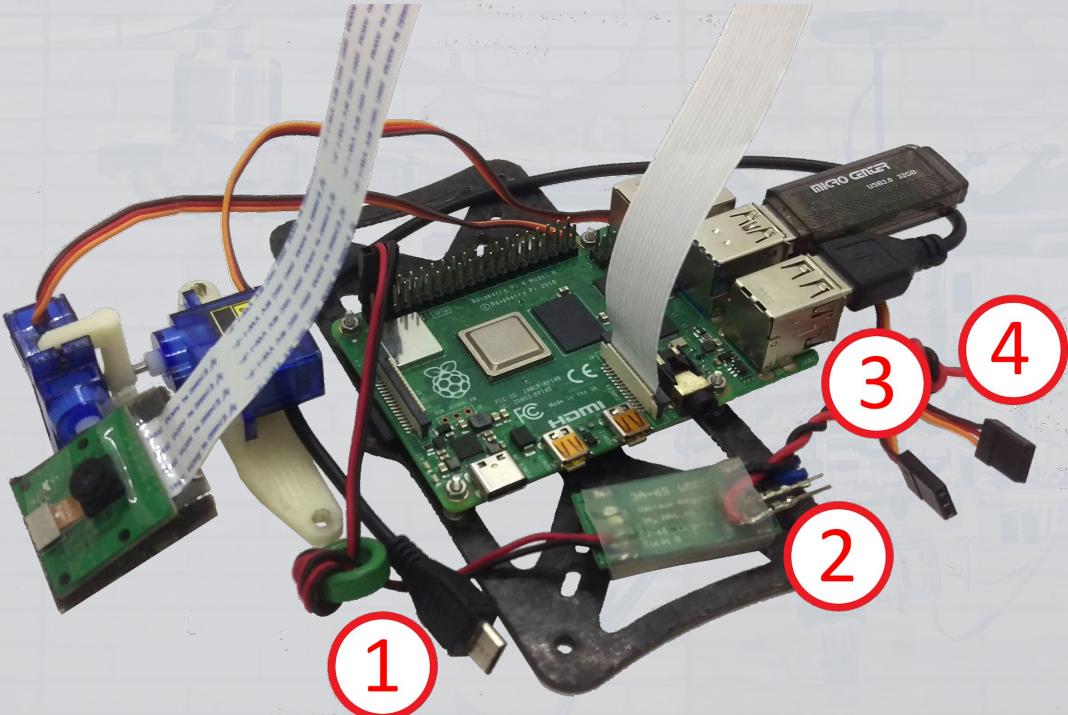
## → Economic

Providing an open source computer vision drone platform at a price point of less than \$1000 makes it possible for individual developers to acquire and prototype different variants for military use

# Hardware:



# Computer Vision Package



## Interface:

1. Pixhawk interface
2. 5V UBEC
3. Pitch servo lead
4. Roll servo lead

# Flight Controller Selection:

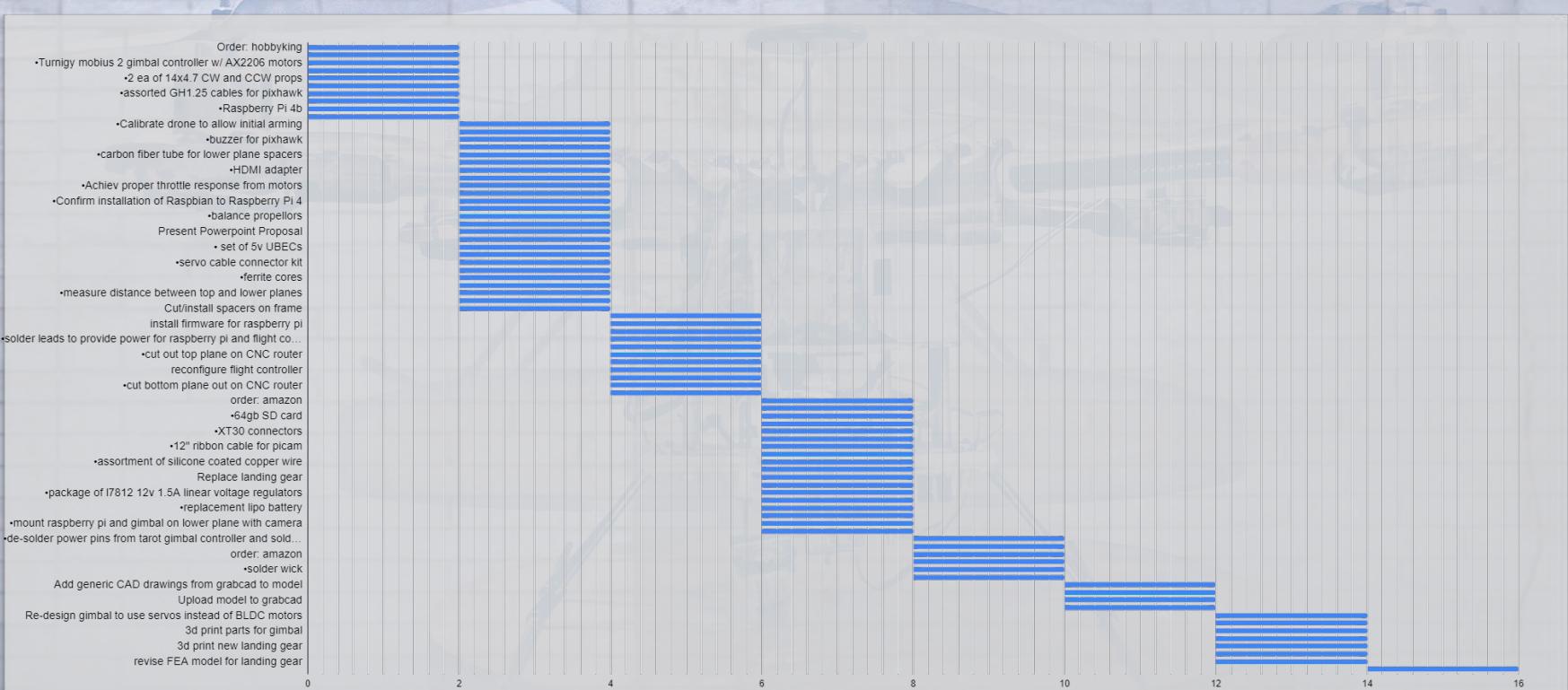
The following flight controllers were considered for this project and ranked in order to select the ideal one for this project

Flight Controller	Cost	Ease of Use	Support	Features	Total
Pixhawk Px4 V2	5	\$4.00	\$5.00	4	<b>18</b>
Pixhawk 4	3	\$4.00	\$5.00	5	17
ErleBrain	1	\$1.00	\$1.00	4	7
APM 2.8	3	\$4.00	\$4.00	4	15
Betaflight F4	5	\$5.00	\$3.00	3	16

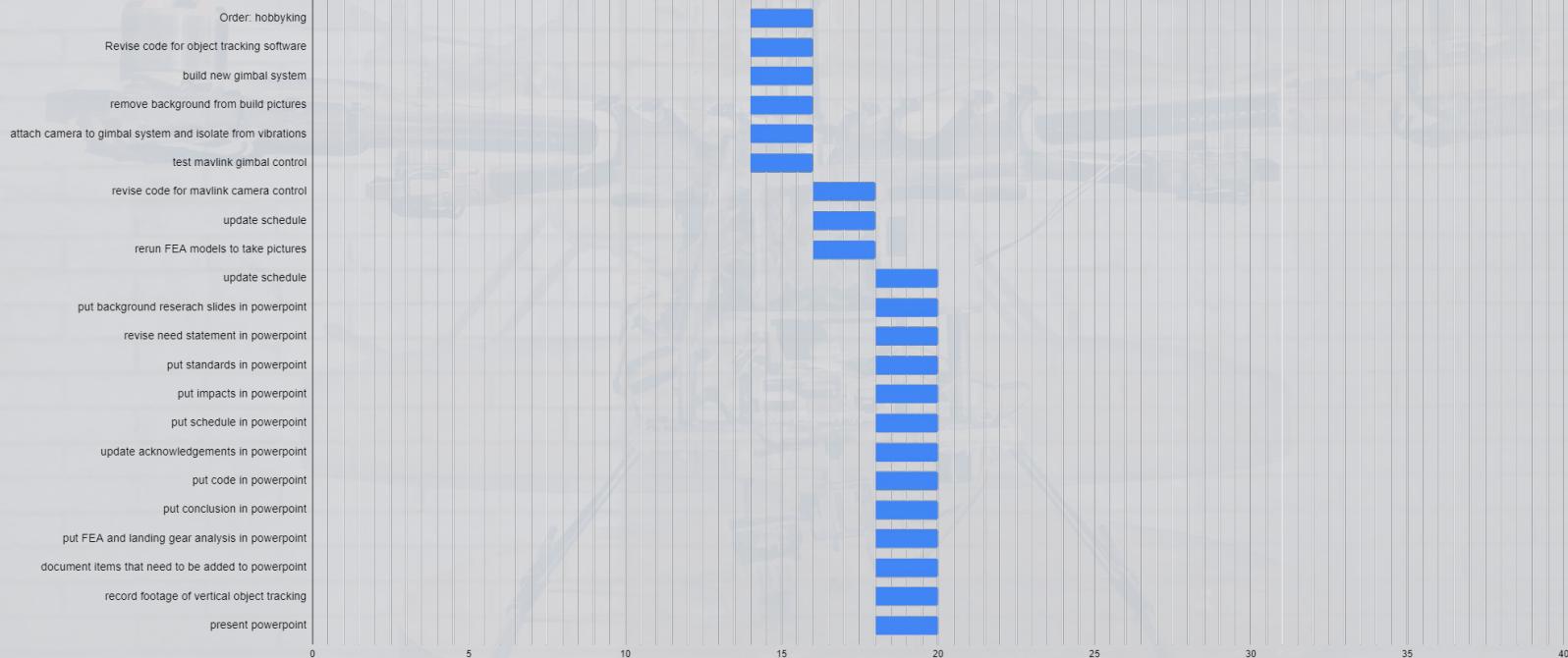
# Development Cost Breakdown

Quantity	Item	Unit Price	Price
1	Raspberry Pi 4B	\$65.00	\$65.00
2	SG90 servo motors	\$7.99	\$15.98
1	Webcam	\$15.00	\$15.00
4	XT30 Connectors	\$0.30	\$1.20
2	XT60 Connectors	\$0.70	\$1.40
	Power Cables	\$0.10	\$0.40
1	Communication Cable	\$0.05	\$0.05
1	Pixhawk PX4 Flight Controller	\$59.35	\$59.35
1	Buzzer	\$7.90	\$7.90
1	5V 2A Battery Eliminator Circuit	\$11.99	\$11.99
1	APM Power Module/Battery Eliminator Circuit	\$6.43	\$6.43
1	4S 3600mAh LiPo Battery	\$61.99	\$61.99
1	3DR Telemetry Transcievers	\$27.99	\$27.99
1	Flysky TH-9X Remote Control	\$87.81	\$87.81
1	D4R-II Reciever	\$24.99	\$24.99
1	Propellor Set	\$6.92	\$6.92
4	50A Electronic Speed Controller	\$26.99	\$107.96
4	BC-35-36 Motors	\$29.99	\$119.96
1	U-blox Neo M8N GPS	\$17.99	\$17.99
1	Carbon Fiber Tube	\$9.66	\$9.66
12	6-32 threaded rods	\$0.18	\$2.16
	Carbon Fiber Sheet		\$0.00
	Carbon Fiber Cloth		\$0.00
	Epoxy Resin		\$0.00
	CNC Machine Time		\$0.00
			\$652.13

# Scheduling

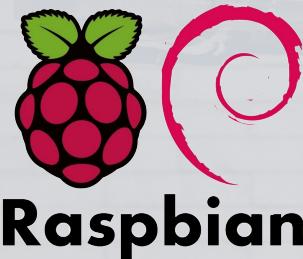


# Scheduling



# Software Stack:

Computer Vision Package



Raspbian



OpenCV



Configuration/Communication

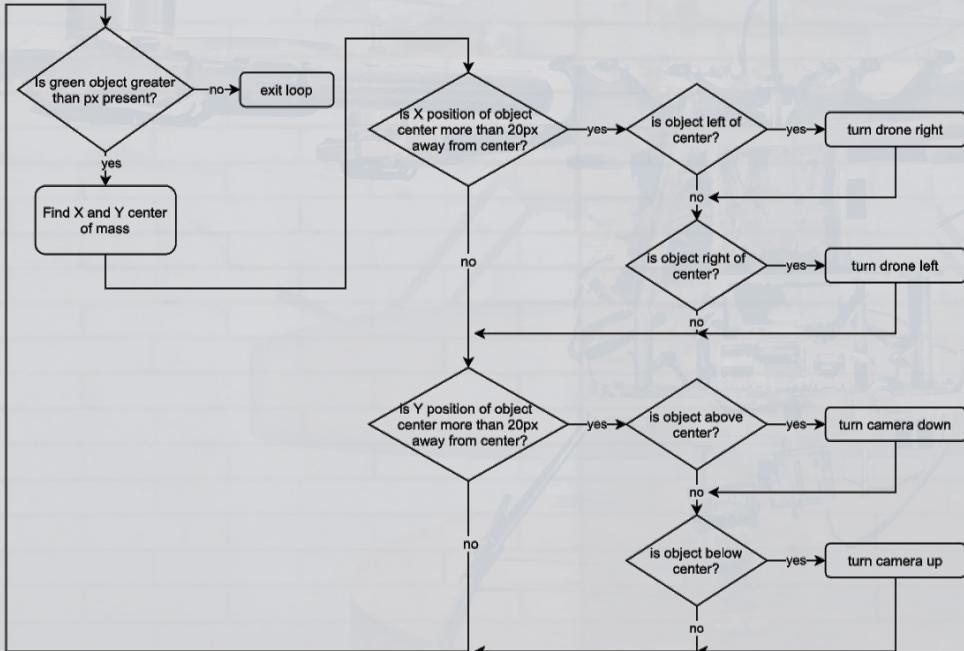


Flight Controller Firmware

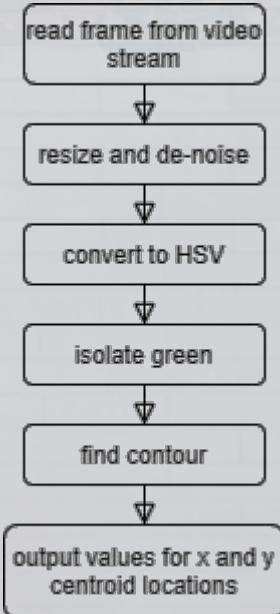


# Object Tracking:

## Control Logic



## OpenCV



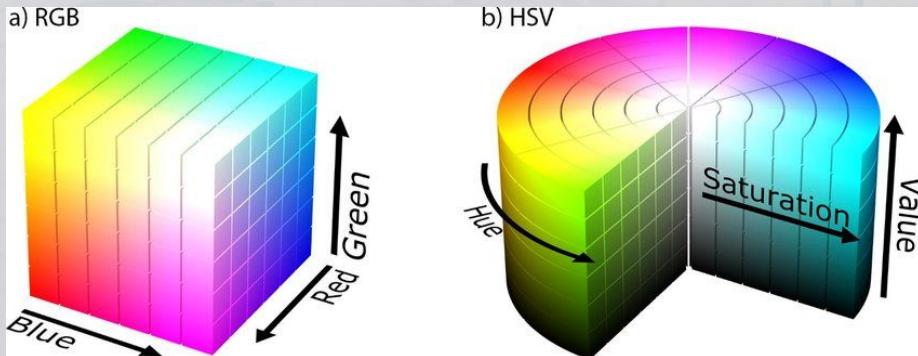
# Code:

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 # import functions for opencv
5 from __future__ import print_function
6 from collections import deque
7 from imutils.video import VideoStream
8 import numpy as np
9 import cv2
10 import imutils
11 import time
12 from dronekit import connect, VehicleMode, LocationGlobal, LocationGlobalRelative
13 from pymavlink import mavutil # Needed for command message definitions
14 import math
15
16 # variables for camera/gimbal settings
17 x_max = 600 # width of pi camera image
18 y_max = 450 # height of pi camera image
19 x_center = x_max / 2 # center value used for readability/ease of editing
20 y_center = y_max / 2 # center value used for readability/ease of editing
21 gimbal_y_pos = -45 # goes from -90 to 0
22 pix_per_deg = 40 # used to adjust movement sensitivity
23
24 def mapObjectPosition(x, y):
25     # print object coordinates
26     print("[INFO] Object Center coordinates at X0 = {} and Y0 = {}".format(x, y))
27
28 greenLower = (29, 86, 6) # define lower boundaries of green
29 greenUpper = (64, 255, 255) # define upper boundaries of green
30 pts = deque(maxlen=64) # list of tracked points
31
32 # load the webcam video stream
33 vs = VideoStream(usePiCamera=True).start()
34
35 # serial connection for pixhawk
36 connection_string = '/dev/serial/by-id/usb-Ardupilot_fmuv2_1C0034000951353332373834-if00'
37 print('Connecting to vehicle on: %s' % connection_string)
38 vehicle = connect(connection_string, wait_ready=True) # Connect to the Vehicle
39 vehicle.flush() # send a "clear all" type message
40
```

```
41 # keep looping
42 while True:
43     # grab the current frame
44     frame = vs.read()
45
46     # handle the frame from VideoCapture or VideoStream
47     frame = frame[1] if args.get("video", False) else frame
48
49     # resize the frame
50     frame = imutils.resize(frame, width=600)
51     # blur it
52     blurred = cv2.GaussianBlur(frame, (11, 11), 0)
53     # convert to HSV color space
54     hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)
55
56     # construct a mask for the color "green"
57     mask = cv2.inRange(hsv, greenLower, greenUpper)
58     # filter false positives
59     mask = cv2.erode(mask, None, iterations=2)
60     #filter false negatives
61     mask = cv2.dilate(mask, None, iterations=2)
62
63     # find contours in the mask and initialize the current
64     # (x, y) center of the ball
65     cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
66                            cv2.CHAIN_APPROX_SIMPLE)
67     cnts = imutils.grab_contours(cnts)
68     center = None
69
70     # only proceed if at least one contour was found
71     if len(cnts) > 0:
72         # find the largest contour in the mask
73         c = max(cnts, key=cv2.contourArea)
74         # compute the minimum enclosing circle
75         ((x, y), radius) = cv2.minEnclosingCircle(c)
76         M = cv2.moments(c) # find centroid
77         center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
78
79         # only proceed if the radius meets a minimum size
80         if radius > 10:
```

# Image Processing:

## Colorspace



**RGB:** Interpolation between light and dark shades requires all values to change

**HSV:** Interpolation between light and dark shades requires only "Value" value to change. More tolerant of variable lighting conditions

# Image Processing:

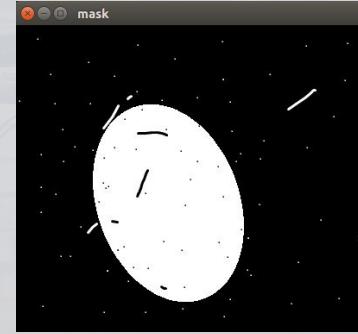
## Gaussian Blur

$$\frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Pixels are averaged against their neighbors proportionately according to a gaussian distribution of weights per pixel distance

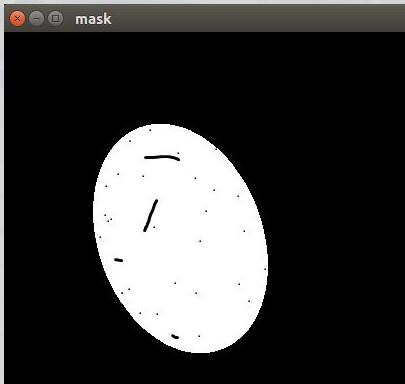
## Mask



Filters out values which do not fall within the specified range.

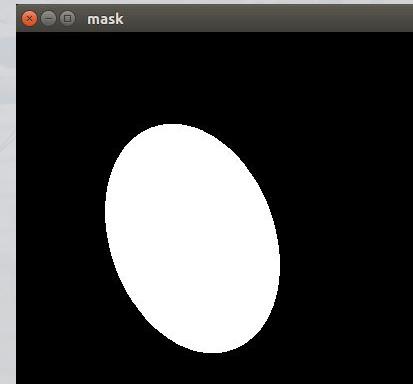
# Image Processing:

## Erode Filter



Removes false positives due to camera noise and other artifacts

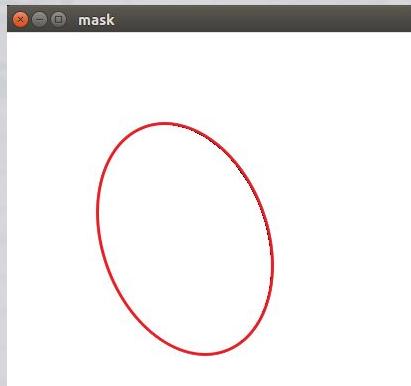
## Dilate Filter



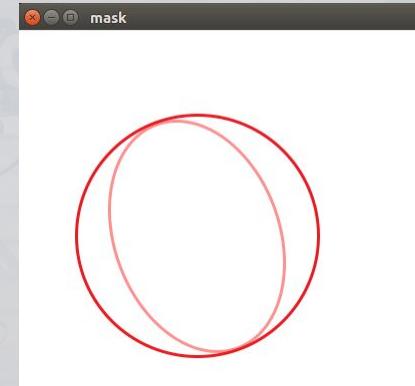
Removes false negatives which fall within the bounds of the mask boundaries

# Image Processing:

## Grab Contour



## Find Enclosing Circle

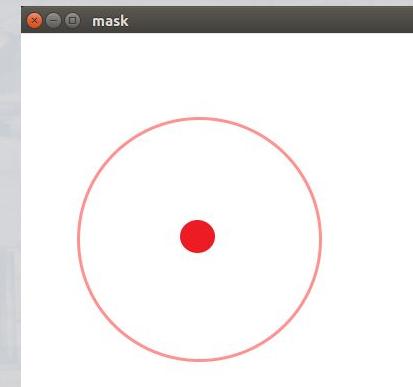
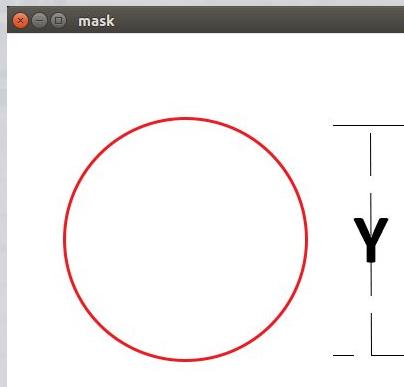


Finds contour based on the edge boundary between the masked and unmasked portions of the image

Finds the smallest circle which encloses the contour

# Image Processing:

## Find Centroid



Finds centroid based on mean area distribution in the enclosed circle

# Code:

```
81     # draw the circle
82     cv2.circle(frame, (int(x), int(y)), int(radius),
83                 (0, 255, 255), 2)
84     # draw the centroid
85     cv2.circle(frame, center, 5, (0, 0, 255), -1)
86     # update the list of tracked points
87     mapObjectPosition(int(x), int(y))
88
89     x = x_max - x # flip so zero = left
90     y = y_max - y # flip so zero = bottom
91
92     # adjust gimbal to follow y centroid
93     if (y > (y_center + pix_per_deg)):
94         print("above")
95         gimbal_y_pos = gimbal_y_pos + (y - y_center)/pix_per_deg
96     elif (y < (y_center - pix_per_deg)):
97         print("below")
98         gimbal_y_pos = gimbal_y_pos - (y_center - y)/pix_per_deg
99     if gimbal_y_pos > 0:
100        gimbal_y_pos = 0
101    elif gimbal_y_pos < -90:
102        gimbal_y_pos = -90
103    vehicle.gimbal.rotate(int(gimbal_y_pos), 0, 0)
104
105    # update the points queue
106    pts.appendleft(center)
107
108    # loop over the set of tracked points
109    for i in range(1, len(pts)):
110        # ignore non-valid points
111        if pts[i - 1] is None or pts[i] is None:
112            continue
113
114        # Compute thickness of line
115        thickness = int(np.sqrt(64 / float(i + 1)) * 2.5)
116        # draw the connecting lines
117        cv2.line(frame, pts[i - 1], pts[i], (0, 0, 255), thickness)
118
119    # show the frame to our screen
120    cv2.imshow("Frame", frame)
```

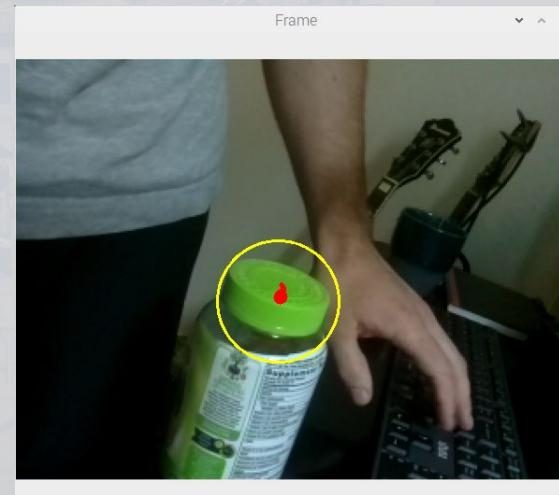
```
121     key = cv2.waitKey(1) & 0xFF
122
123     # if the 'q' key is pressed, stop the loop
124     if key == ord("q"):
125         break
126
127     # Close vehicle object
128     print("Close vehicle object")
129     vehicle.close()
130
131     # stop the camera video stream
132     vs.stop()
133
134     print("Completed")
```

# Image Processing:

Centroid values output

```
pi@raspberrypi: ~/Desktop/fromgit
File Edit Tabs Help
[INFO] Object Center coordinates at X0 = 281 and Y0 = 252
[INFO] Object Center coordinates at X0 = 281 and Y0 = 252
[INFO] Object Center coordinates at X0 = 284 and Y0 = 250
[INFO] Object Center coordinates at X0 = 283 and Y0 = 251
[INFO] Object Center coordinates at X0 = 282 and Y0 = 251
[INFO] Object Center coordinates at X0 = 282 and Y0 = 250
[INFO] Object Center coordinates at X0 = 282 and Y0 = 250
[INFO] Object Center coordinates at X0 = 280 and Y0 = 252
[INFO] Object Center coordinates at X0 = 280 and Y0 = 252
[INFO] Object Center coordinates at X0 = 281 and Y0 = 252
[INFO] Object Center coordinates at X0 = 280 and Y0 = 252
[INFO] Object Center coordinates at X0 = 280 and Y0 = 252
[INFO] Object Center coordinates at X0 = 280 and Y0 = 253
[INFO] Object Center coordinates at X0 = 280 and Y0 = 253
[INFO] Object Center coordinates at X0 = 280 and Y0 = 253
[INFO] Object Center coordinates at X0 = 281 and Y0 = 257
[INFO] Object Center coordinates at X0 = 281 and Y0 = 257
[INFO] Object Center coordinates at X0 = 281 and Y0 = 258
[INFO] Object Center coordinates at X0 = 281 and Y0 = 258
[INFO] Object Center coordinates at X0 = 281 and Y0 = 257
[INFO] Object Center coordinates at X0 = 281 and Y0 = 258
[INFO] Object Center coordinates at X0 = 281 and Y0 = 258
[INFO] Object Center coordinates at X0 = 280 and Y0 = 259
```

Enclosing circle and centroid trace



# Object Tracking:

## Gimbal Trim Sensitivity

- 40px/deg value for close-range
- $|\text{Centroid} - \text{y center}| > 40$ :
  - Rotate drone  $|\text{Centroid} - \text{y center}|/40$  degrees left/right
- To track objects in variable ranges:
  - Store values for: camera position, object centroid distance, time
  - Noise-filter
  - Transfer to 2D Numpy array
  - Increase px\_per\_deg value: if overshoot detected
  - Decrease px\_per\_deg value: if response is lagging

# Object Tracking:

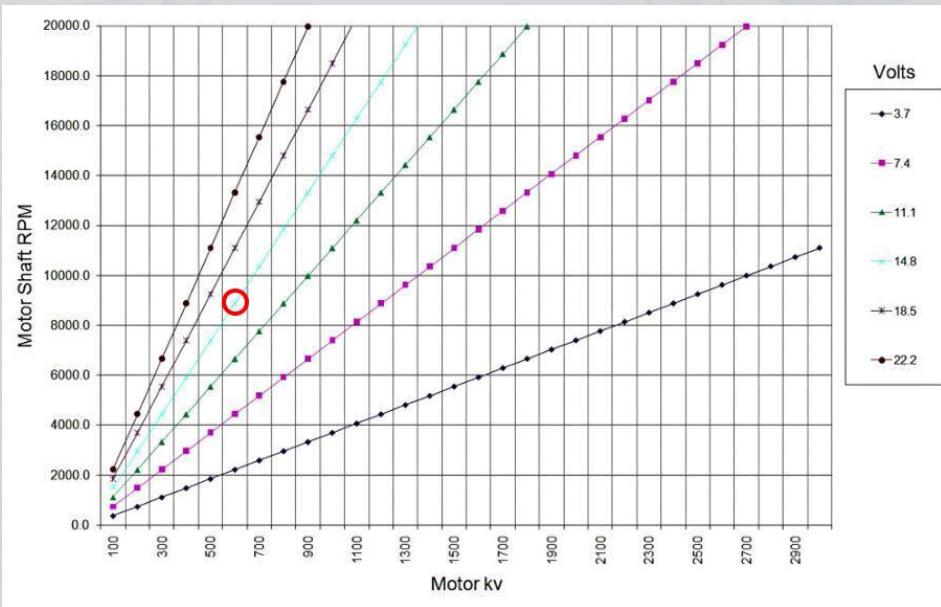
## Yaw Trim Sensitivity

- 40px/deg value for close-range
- $|\text{Centroid} - \text{y center}| > 40$ :
  - Move camera  $|\text{Centroid} - \text{y center}|/40$  degrees up/down between -90/0 degrees
- To track objects in variable ranges:
  - Store values for: camera position, object centroid distance, time
  - Noise-filter
  - Transfer to 2D Numpy array
  - Increase px\_per\_deg value: if overshoot detected
  - Decrease px\_per\_deg value: if response is lagging

# Object Tracking:



# Thrust Calculations:

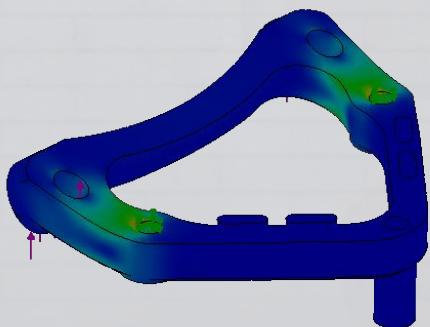
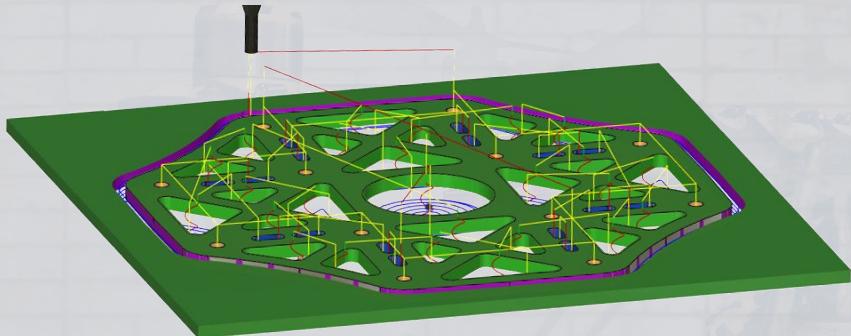


Estimated 8880 RPM maximum speed

1. *UAV Weight = 6.4lbs*
2. *Static Thrust(N):*  
$$1.86 \times 10^{-11} (RPM^2 * \sqrt{Pitch} * D^{3.5})$$
3. *Known Variables:*
  - a.  $RPM = 8880$
  - b.  $D = 14in$
  - c.  $Pitch = 4.7in$
  - d.  $V_0 = 0m/s$
4. *Static Thrust = 32.6N or 7.2lbf*
5. *Thrust to Weight Ratio = 1.13*

# Development Lessons Learned:

Simulate



Simplify!



# Development Lessons Learned:



## Original Gimbal:

- Heavy
- Allows for PID control
- Dedicated motor drivers
- Requires balancing
- Requires dedicated software
- Bulky

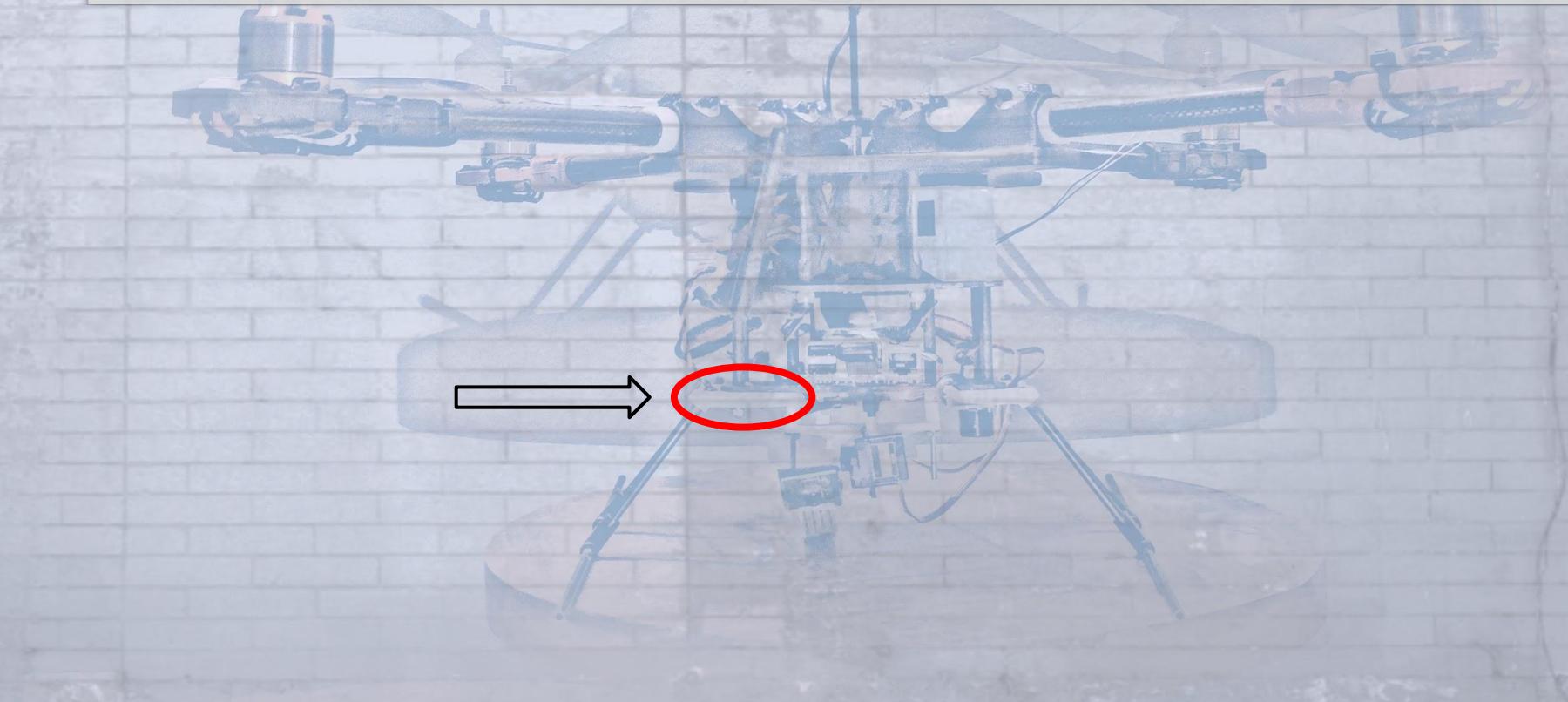
# Development Lessons Learned:



## Updated Gimbal:

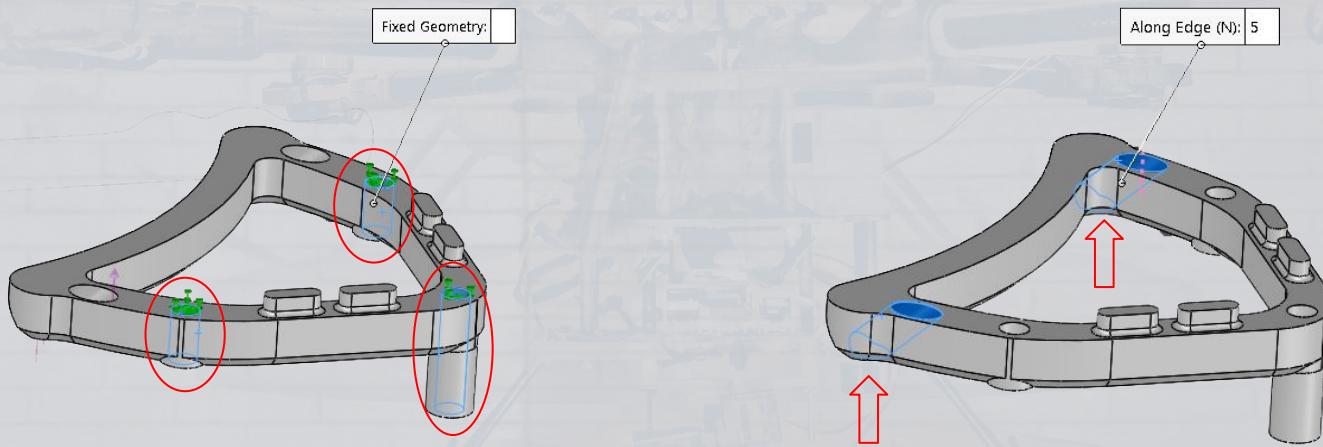
- Light
- Driven by flight controller
- Configurable through flight controller software

# Designing the Landing Gear... for the Drone... for the Object Tracking System:



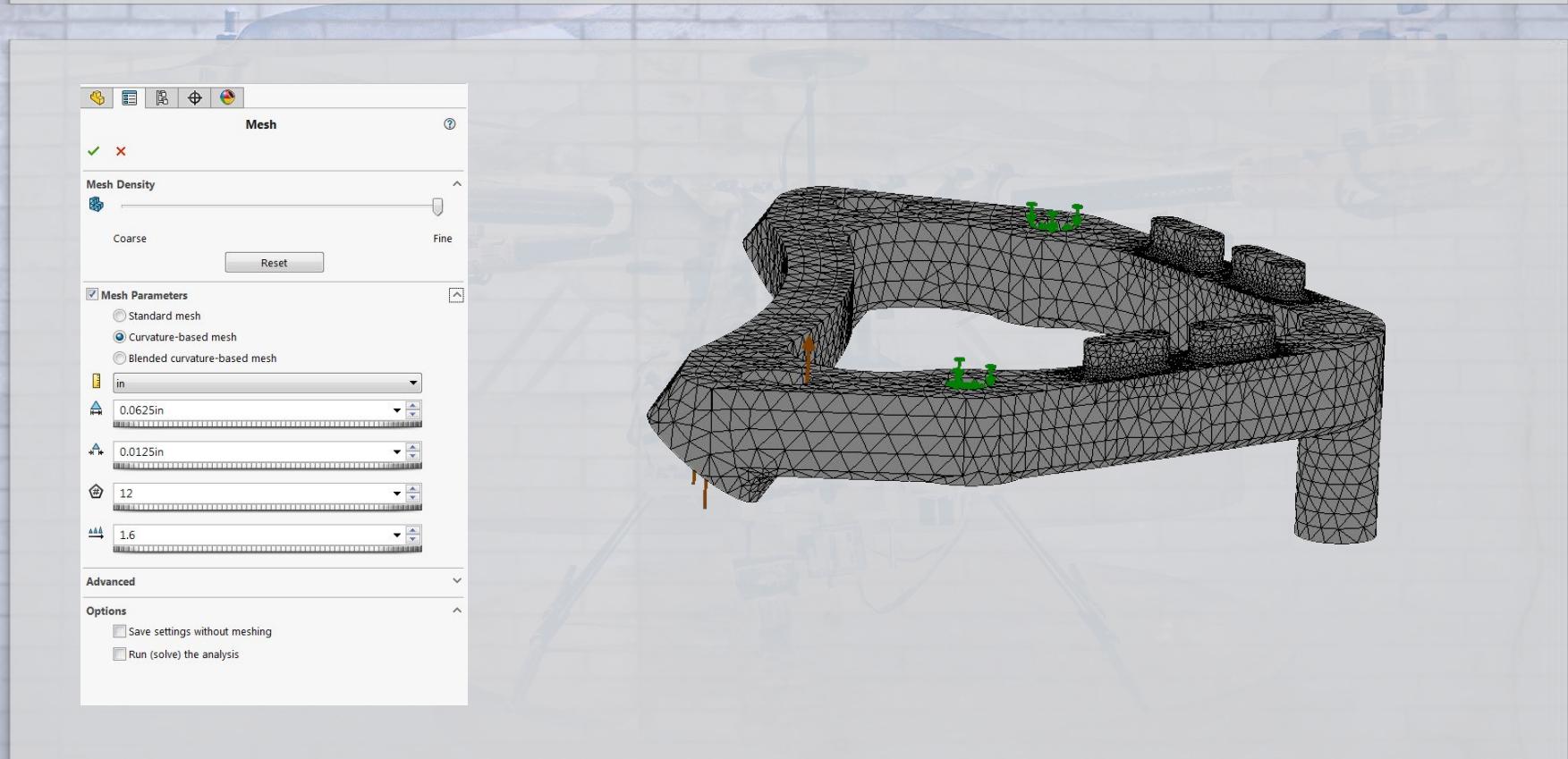
# Methodology:

Iterative Design Process, Isolate 1 Variable to Change

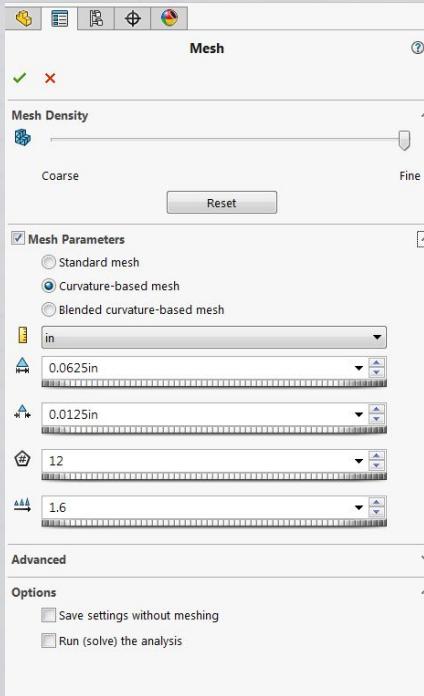


Consistent Loading across iterations

# Creating a Representative Mesh Model



# Creating a Representative Mesh Model



**Maximum Tetrahedron Size: 1/16"**

**Minimum Tetrahedron Size: 1/80"**

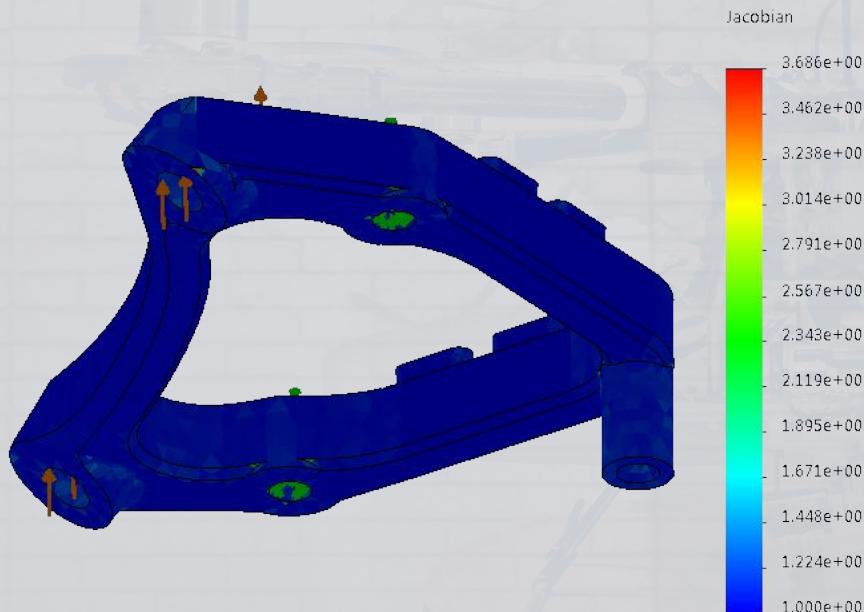
**Minimum Edges/Circle: 12**

**Maximum Curvature/Edge: 30deg**

**Maximum Adjacent Tetrahedron  
Growth Rate: +60%**

**Avg Simulation Time: 3.5 minutes**

# Creating a Representative Mesh Model



## Jacobian Mesh Quality:

Mesh resolution increased until high value instances were no longer found in load path

# Values Chosen for Loading:

Initial drone expected weight: 2kg

Number of landing gear legs: 4

Force per leg =  $(9.81\text{m/s}^2 \times 2\text{kg})/4 \approx 5\text{N}$

Marin Factors: Not feasible to Use

- **K<sub>a</sub>** (surface factor) - Not available for PLA or Epoxy Resin
- **K<sub>b</sub>** (size factor) - No model available for given cross section
- **K<sub>c</sub>** (loading factor) = 1 (in bending)
- **K<sub>d</sub>** (temperature factor) - Varies widely for PLA and Epoxy

# Material Properties:

## Epoxy

The screenshot shows the SOLIDWORKS Material Properties dialog for Epoxy. The left sidebar lists material categories like SOLIDWORKS DIN Materials, SOLIDWORKS Materials (Steel, Iron, Alloys, Plastics), and specific materials like ABS, Epoxy, and Delrin. The main panel has tabs for Properties, Tables & Curves, Appearance, CrossHatch, Custom, and Application Data. The Properties tab is active, showing the following details:

**Material properties**  
Materials in the default library can not be edited. You must first copy the material to a custom library to edit it.

**Model Type:** Linear Elastic Isotropic  
**Units:** SI - N/m<sup>2</sup> (Pa)  
**Category:** Plastics  
**Name:** Epoxy, Unfilled  
**Default failure criterion:** Unknown  
**Description:** Epoxy, Unfilled  
**Source:**   
**Sustainability:** Defined

**Property** **Value** **Units**

Shear Modulus		N/m <sup>2</sup>
Mass Density	1100	kg/m <sup>3</sup>
Tensile Strength	2800000	N/m <sup>2</sup>
Compressive Strength	104000000	N/m <sup>2</sup>
Yield Strength		N/m <sup>2</sup>
Thermal Expansion Coefficient		/K
Thermal Conductivity	0.188	W/(m·K)
Specific Heat		J/(kg·K)
Material Damping Ratio		N/A

Buttons at the bottom: Apply, Close, Save, Config..., Help.

## Polylactic Acid (PLA)

The screenshot shows the SOLIDWORKS Material Properties dialog for PLA. The left sidebar lists material categories like PP Film, PS Medium/High Flow, PVA, PVB, PVC, Sheet Moulding Compound, SMA, and Sustainability Extras. The main panel has tabs for Properties, Tables & Curves, Appearance, CrossHatch, Custom, and Application Data. The Properties tab is active, showing the following details:

**Material properties**  
Materials in the default library can not be edited. You must first copy the material to a custom library to edit it.

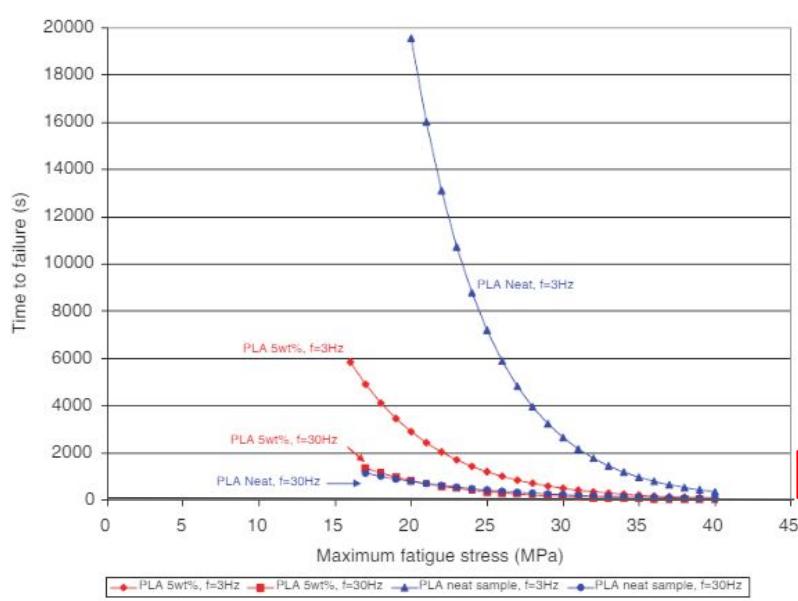
**Model Type:** Linear Elastic Isotropic  
**Units:** SI - N/m<sup>2</sup> (Pa)  
**Category:** Plastic  
**Name:** PLA  
**Default failure criterion:** Max von Mises Stress  
**Description:** -  
**Source:**   
**Sustainability:** Undefined

**Property** **Value** **Units**

Elastic Modulus	200000000	N/m <sup>2</sup>
Poisson's Ratio	0.33	N/A
Shear Modulus	240000000	N/m <sup>2</sup>
Mass Density	1290	kg/m <sup>3</sup>
Tensile Strength	30500000	N/m <sup>2</sup>
Compressive Strength	40000000	N/m <sup>2</sup>
Yield Strength	30000000	N/m <sup>2</sup>
Thermal Expansion Coefficient		/K
Thermal Conductivity	0.2256	W/(m·K)

Buttons at the bottom: Apply, Close, Save, Config..., Help.

# Endurance Limit:



For less than 100 expected cycles:

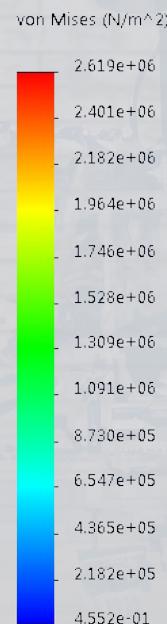
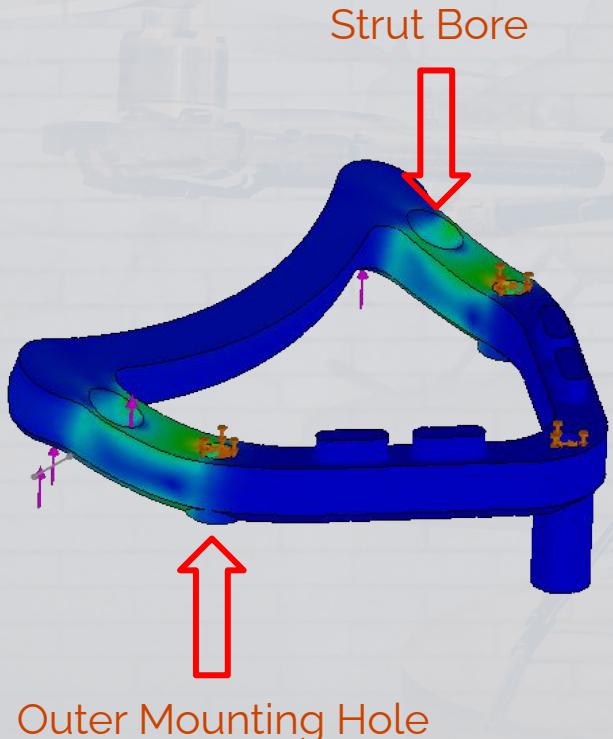
$$100 \text{ cycles} \times 3\text{hz} = 33\text{s}$$

3s

Endurance limit is not relevant!

\*Mechanical Behaviour of Poly(Lactic Acid)

# 1st Iteration: Epoxy



**Epoxy yield strength:**

$$6.5 \times 10^7 \text{ N/m}^2$$

**Factor of Safety:**

$$\text{F.O.S} = (6.5 \times 10^7 \text{ N/m}^2) / (2.6 \times 10^6 \text{ N/m}^2)$$

$$\text{F.O.S} = 25$$

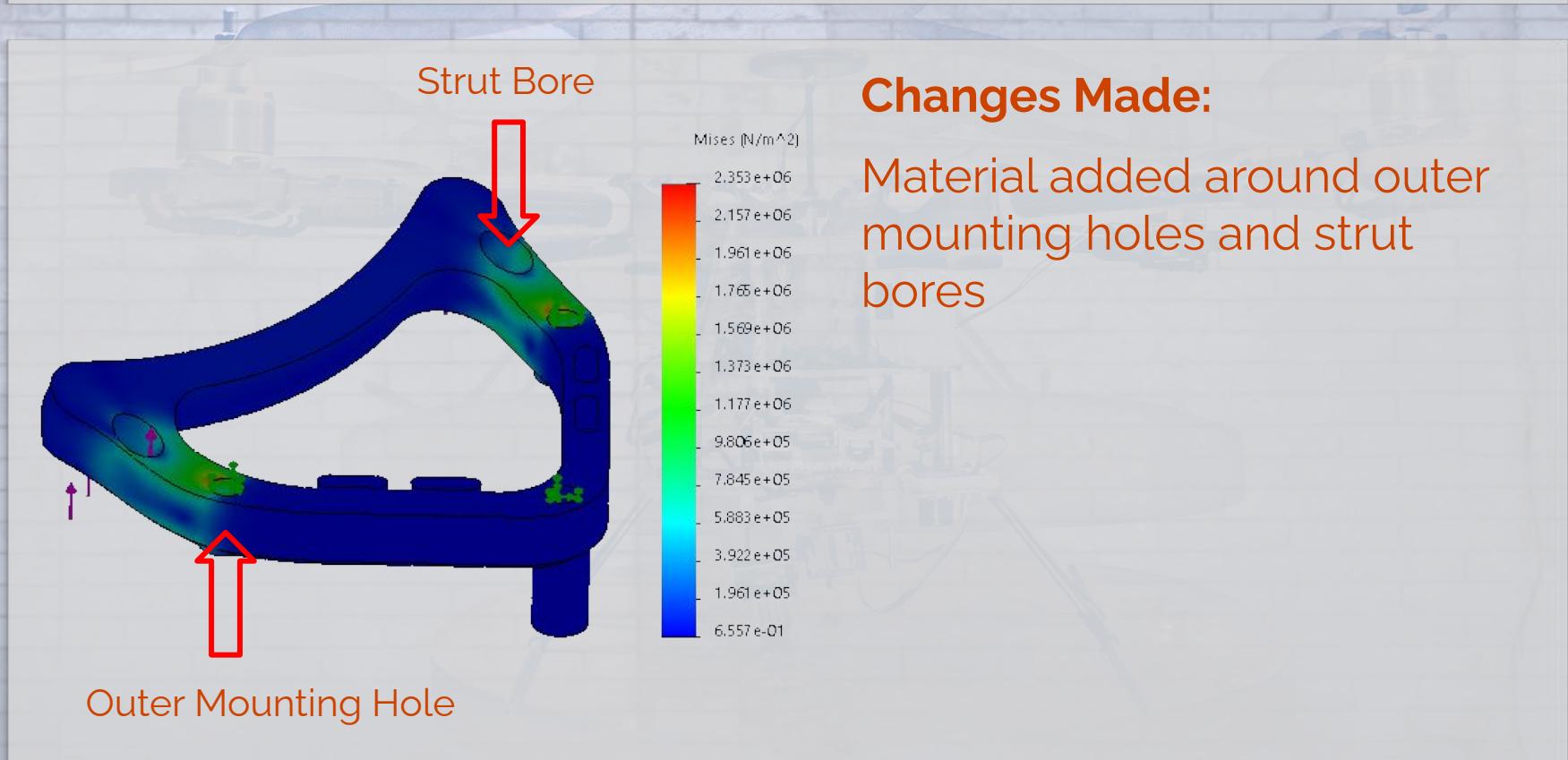
**Location of Highest Stress:**

Outer Mounting Holes

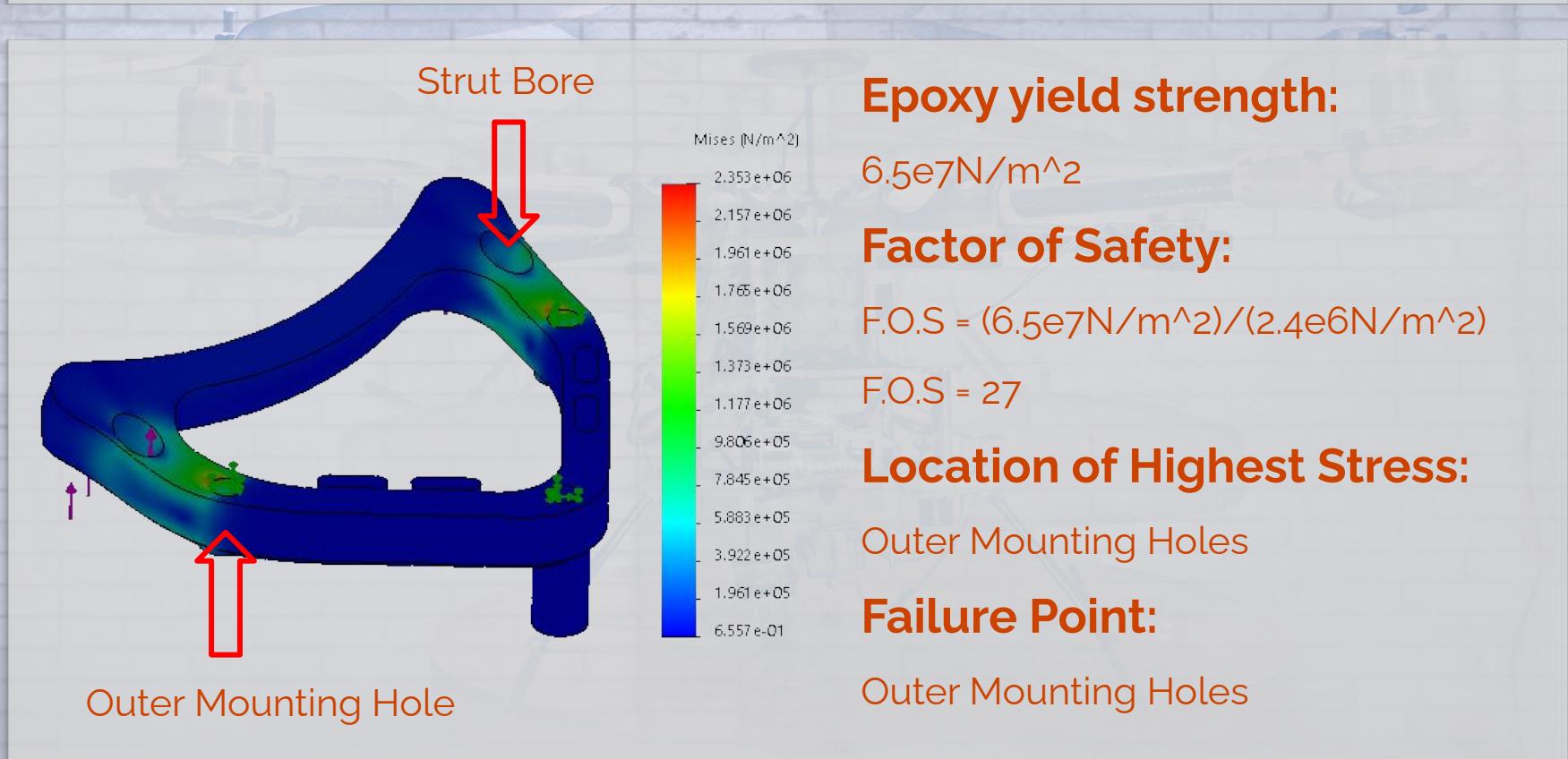
**Failure Point:**

Strut Bore Wall

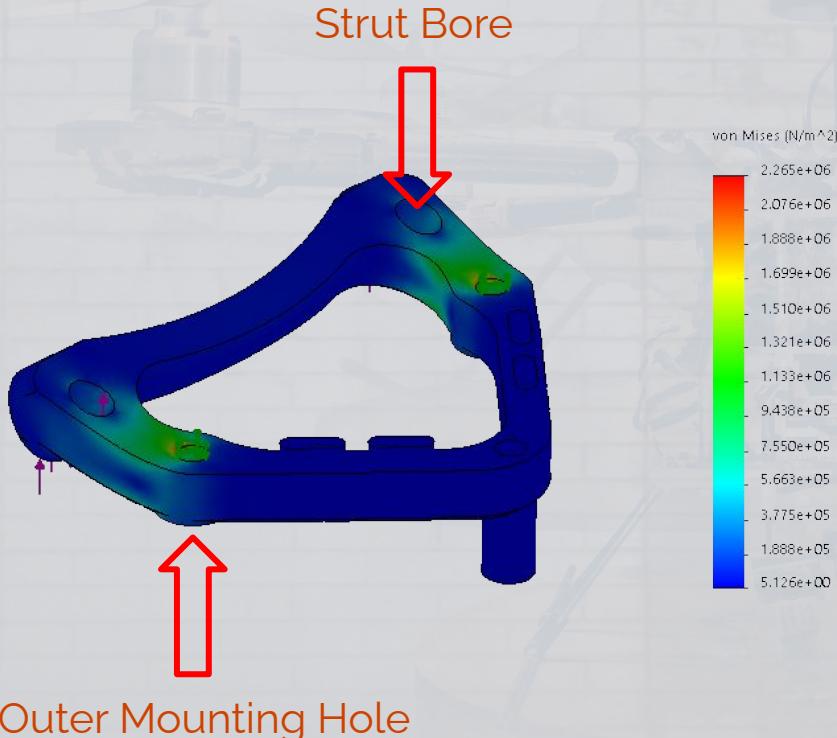
# 2nd Iteration: Epoxy



# 2nd Iteration: Epoxy



# 3rd Iteration: Epoxy

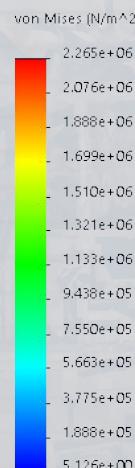
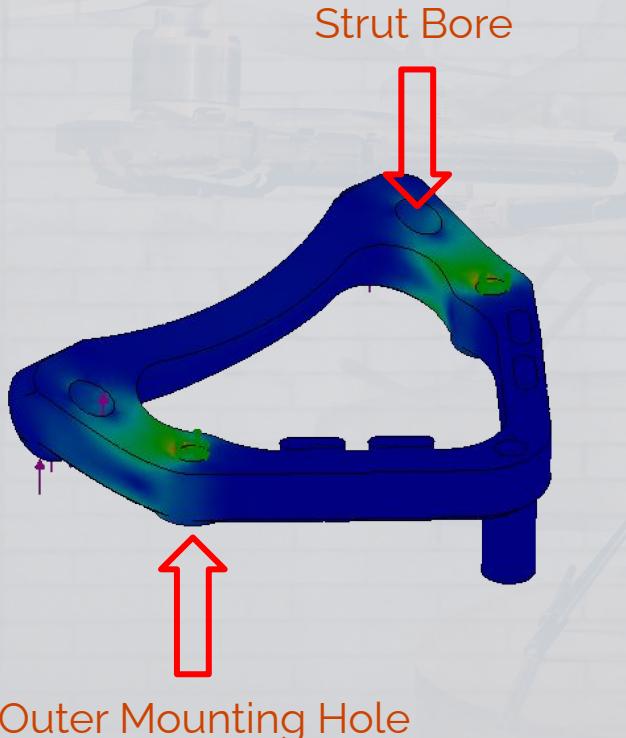


## Changes Made:

Distance between strut bore and outer mounting hole decreased

Material added around mounting hole

# 3rd Iteration: Epoxy



**Epoxy yield strength:**

$6.5 \times 10^7 \text{ N/m}^2$

**Factor of Safety:**

$$\text{F.O.S.} = (6.5 \times 10^7 \text{ N/m}^2) / (2.3 \times 10^6 \text{ N/m}^2)$$

$$\text{F.O.S.} = 27$$

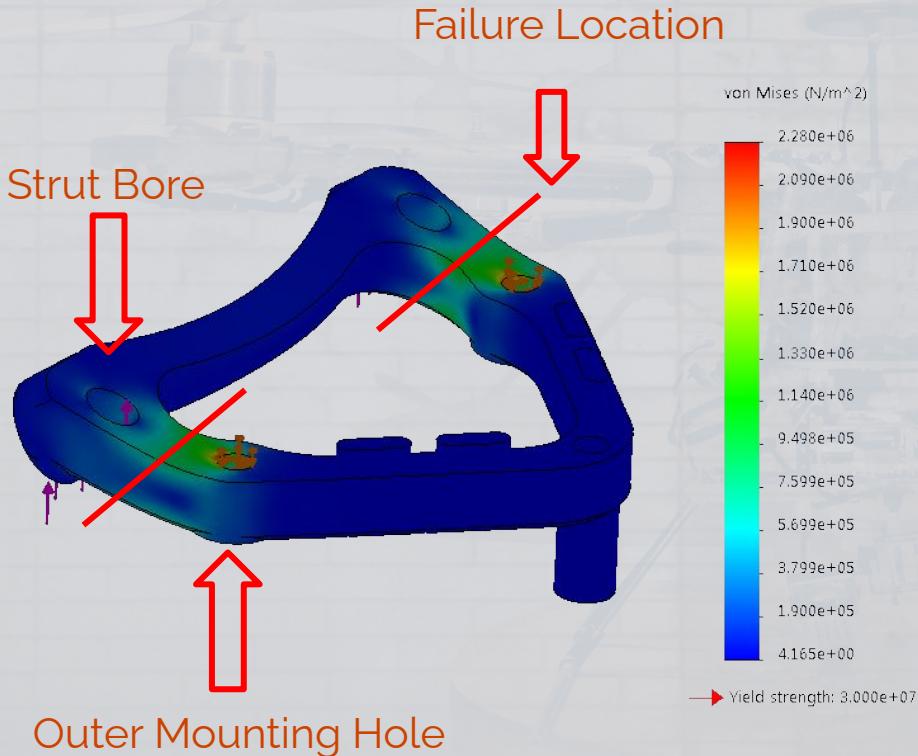
**Location of Highest Stress:**

Outer Mounting Holes

**Failure Point:**

Strut Bore Wall

# 3rd Iteration: PLA

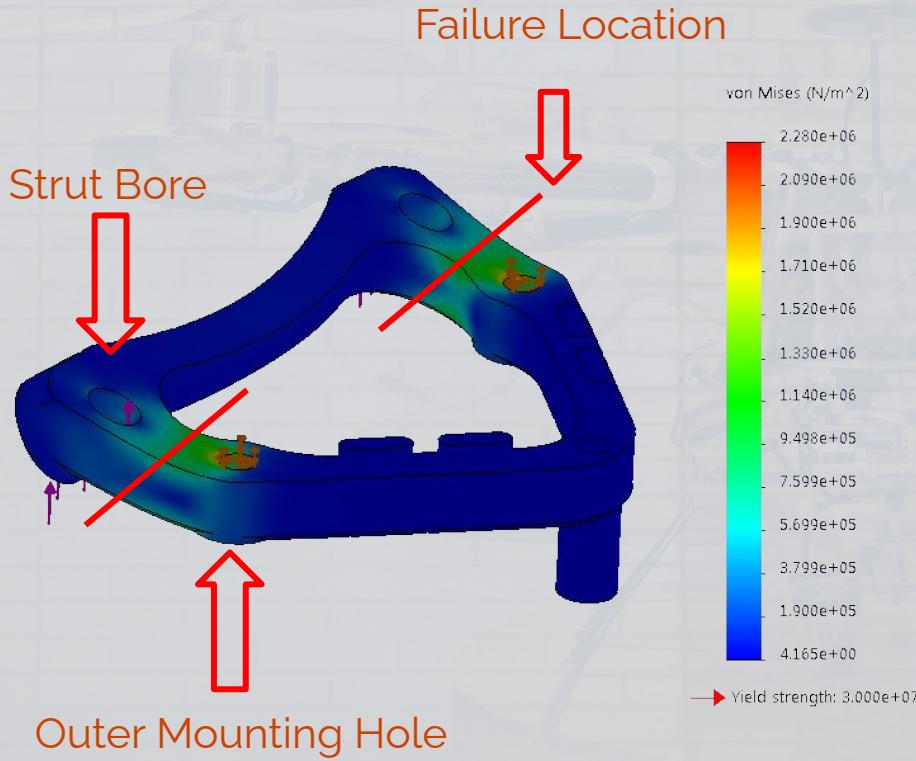


## Changes Made:

Material changed to PLA

Struts shortened from 5" to 3.5" to reduce potential torsional load

# 3rd Iteration: PLA



**PLA yield strength:**

$$5.6 \times 10^7 \text{ N}/\text{m}^2$$

**Factor of Safety:**

$$\text{F.O.S.} = (5.6 \times 10^7 \text{ N}/\text{m}^2) / (2.2 \times 10^6 \text{ N}/\text{m}^2)$$

$$\text{F.O.S.} = 25$$

**Location of Highest Stress:**

Outer Mounting Holes

**Failure Point:**

Between Mounting Holes and Strut  
Bores

# Discussion of Iterations



Each iteration proved to be more durable than the last, with the 3rd iteration printed using PLA being the most durable. This is not surprising given the more compliant nature of the material and the brittle nature of epoxy.

Assuming a linearly elastic orthotropic material and a perfectly constrained loading scenario one could expect the PLA bracket to withstand roughly  $\frac{1}{2}$  the force of the resin bracket even though it is stronger. One must consider real-world factors in design. While glass (yield strength =  $7e9 \text{ N/m}^2$ ) would make a superior part if only considering yield strength and the expected forces/modifying factors it would make a poor material choice for complex mechanical structures

# Material Selection

## Epoxy:

- Slow to Print
- Brittle
- Requires Post Cure
- Requires Manual Support Removal
- Requires Cleaning

## Polylactic Acid (PLA):

- Reasonable Print Times
- Easier Support Removal
- Potentially Less Weight
- More Durable

# Finite Element Analysis Takeaways

## $(\sigma_{Yield})/(\sigma_{Max(static)})$ Analysis:

- Favors brittle materials
- Does not produce meaningful data in and of itself
- Useful as a comparative tool between iterations
- Only effective for comparing same material
- Oversimplifies real world loading scenarios
- Can produce disproportionately large  $\sigma_{Max(static)}$  values adjacent to load/fixture locations

# Finite Element Analysis: Improvements

The model could be improved by:

- Adding 3d print layer orientation
- Accounting for orthotropic properties of 3d printed materials
- Performing real world testing to determine properties of parts produced from specific printer and filament/resin for given print/post-processing and environmental settings.  
Published values vary significantly
- Create non-linear elastic profile for non-brittle materials

# Weight:Thrust Reduction:

Updated servo-based gimbal design: new weight (sans camera) was found to be 102g. Previous gimbal design weighed 537g; this saved 435g or 77%.

- Gimbal V1 weight = 537g
- Gimbal V2 weight = 102g
- Weight reduction =  $1 - \frac{\text{Gimbal V1 weight}}{\text{Gimbal V2 weight}}$
- Weight reduction =  $1 - \frac{537g}{102g}$
- Weight reduction = 0.765

Factoring the weight of the drone in (without the gimbal):

- Drone weight = 982g
- Drone weight + Gimbal V1 weight = 982g + 537g
- Drone weight + Gimbal V2 weight = 982g + 102g
- Overall weight reduction =  $1 - \frac{\text{Drone weight} + \text{Gimbal V2 weight}}{\text{Drone weight} + \text{Gimbal V1 weight}}$
- Overall weight reduction =  $1 - \frac{982g + 102g}{982g + 537g}$
- Overall weight reduction =  $1 - \frac{1084g}{1519g}$
- Overall weight reduction = 0.286

29% weight reduction allows for lower load on:

- Propellers
- Motors
- ESCs
- Battery
- Frame

Will allow for:

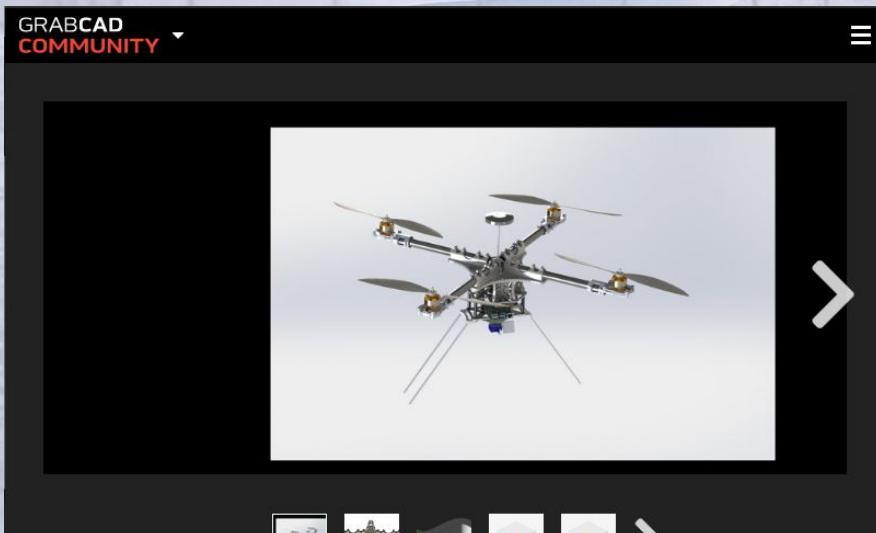
- Increased maneuverability in transient flight conditions
- Increased flight times

# Weight:Thrust Reduction:

## Further Potential Improvements:

- Replace 3d printed components with molded carbon
- Remove from Raspberry Pi: ethernet port, header pins
- Reduce boom vibration isolation material
- Replace 14" FGRP(FiberGlass Reinforced Plastic) propellers with 16" CFRP (Carbon Fiber Reinforced Plastic) propellers
- Produce solid molded carbon motor mounts
- Incorporate landing gear into lower plane
- Source battery with greater power density

# Project Availability:



## Object tracking quadcopter

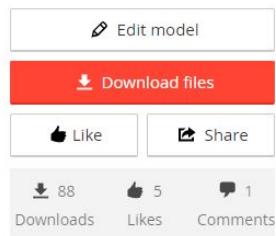


Ryan Lewis

February 3rd, 2020

Quadcopter using pixhawk flight controller, custom designed gimbal and raspberry pi to track objects. See

[https://github.com/rlewis94/Object\\_Tracking\\_Drone](https://github.com/rlewis94/Object_Tracking_Drone) from more details



The image shows a GitHub repository page for 'rlewis94 / Object\_Tracking\_Drone'. The header includes the repository name, a search bar, and navigation links for Pull requests, Issues, Marketplace, and Explore. Below the header is a brief description: 'Drone w/ camera gimbal controlled by Pixhawk and Raspberry Pi running OpenCV'. The repository stats show 10 commits, 1 branch, 0 packages, 0 releases, and 1 contributor. The commit history lists several updates, including 'Update README.md', 'Drone tracking flow chart.drawio', 'Added Drone object tracking flow chart.drawio', 'Update README.md', 'definitions.py', 'Add files via upload', and 'main\_11\_25\_19.py', all made by 'rlewis94' within the last 6 months.

# Conclusion:

- Be flexible and able to pivot in the event that external factors result in the initial project direction no longer being feasible
- The object tracking drone is able to track objects
- Global pandemics increase the difficulty of sourcing parts
- For 3d printed parts:
  - Traditional FEA models provide useful information for iterative design processes
  - Material durabilities cannot be indicated purely by comparing specification sheets. Real world testing is required
- The benefits from documentation and in-depth analysis should be weighed against the amount of time consumed in order to not hinder project development
- Take full accountability for project progress in the event project stakeholders do not take initiative

# Acknowledgements:

- Rachid Nafaa - Manufacturing: CAM profile optimization
- Peter Kalaitzidis - Design: Gimbal implementation feedback
- Alaric Hyland - Manufacturing: 3D printing and material selection feedback
- Dr. Shouling He - Documentation: Feedback
- Vaughn College Engineering Dept - Equipment: 3D printers and router

# References:

- Aircraft Principal Axes. (2019, November 19). Retrieved from [https://en.wikipedia.org/wiki/Aircraft\\_principal\\_axes](https://en.wikipedia.org/wiki/Aircraft_principal_axes)
- HSL and HSV. (2020, April 19). Retrieved from [https://en.wikipedia.org/wiki/HSL\\_and\\_HSV](https://en.wikipedia.org/wiki/HSL_and_HSV)
- PX4. (n.d.). Retrieved from <https://dev.px4.io/master/en/index.html>
- MAVLink. (n.d.). Introduction. Retrieved from <https://mavlink.io/en/>
- OpenCV. (2020, April 09). Retrieved from <https://opencv.org/>
- Chollet, F., Kalbermatter, S., Brownlee, J., Geitgey, A., Malisiewicz, T., Zdziarski, Z., ... IP, P. (2018, September 19). Computer Vision, Deep Learning, and OpenCV. Retrieved from <https://www.pyimagesearch.com/>
- Drone Code. (2020, April 27). Retrieved from <https://www.dronecode.org/>
- Ardupilot. (n.d.). Retrieved from <https://ardupilot.org/>
- Balena. (n.d.). Retrieved from <https://www.balena.io/>
- Averett, Rodney & Realff, Mary & Jacob, Karl & Cakmak, Miko & Yalcin, Baris. (2011). The mechanical behavior of poly(lactic acid) unreinforced and nanocomposite films subjected to monotonic and fatigue loading conditions. *Journal of Composite Materials - J COMPOS MATER.* 45. 2717-2726. 10.1177/0021998311410464.