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:

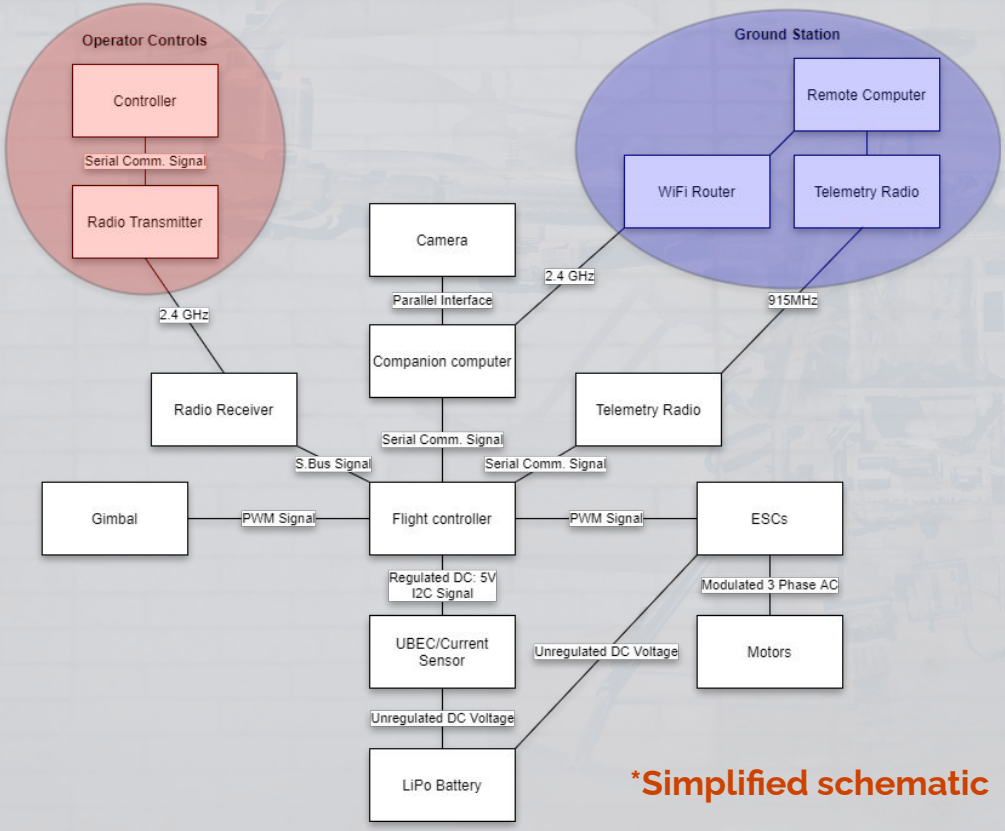
- ➔ **Thrust : Weight Ratio**
  Greater than 1.1
- ➔ **Perform Local Vision Processing**
  Does not require video stream to remote computer for image processing
- ➔ **Track Green Objects**
  Ranging from (29, 86, 6) to (64, 255, 255) in the HSV colorspace

Constraints and Standards:

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

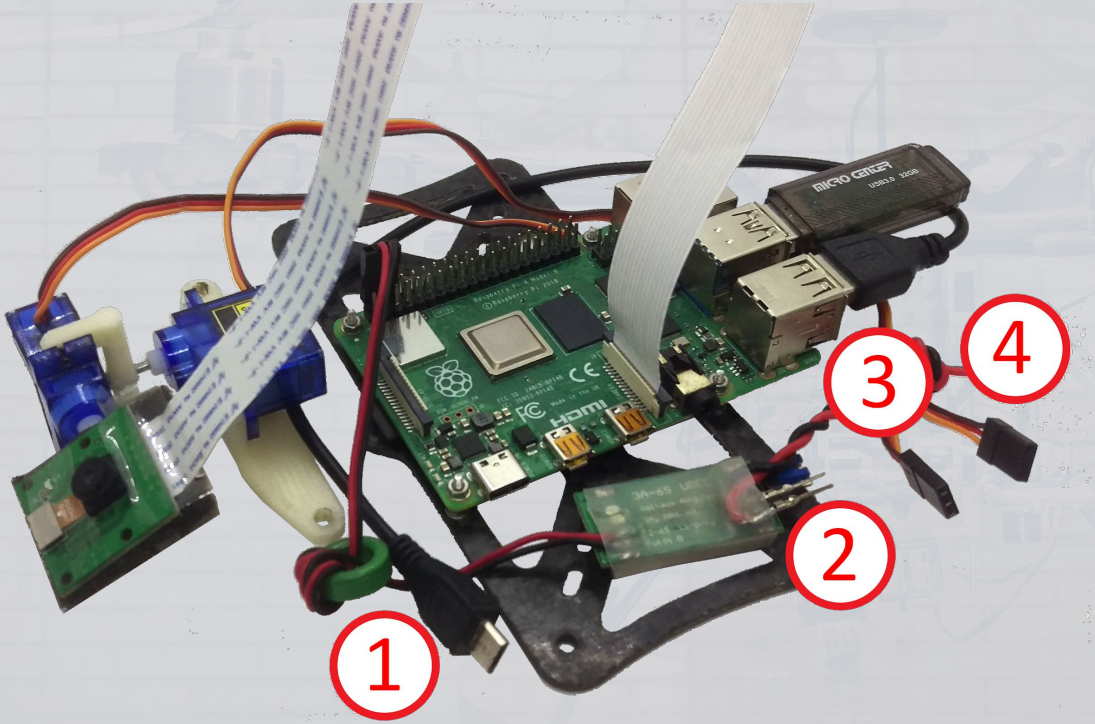
- **IEEE 802.11**
  Wireless LAN communication standards
- **FCC Title 47 C.F.R. § 2.106**
  Frequency allocations: used for telemetry comms
- **Public Law 112-95 Section 336**
  Operating requirements for non-commercial drone use

Hardware:



*Simplified schematic

Computer Vision Package



Interface:

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

Flight Controller Selection:

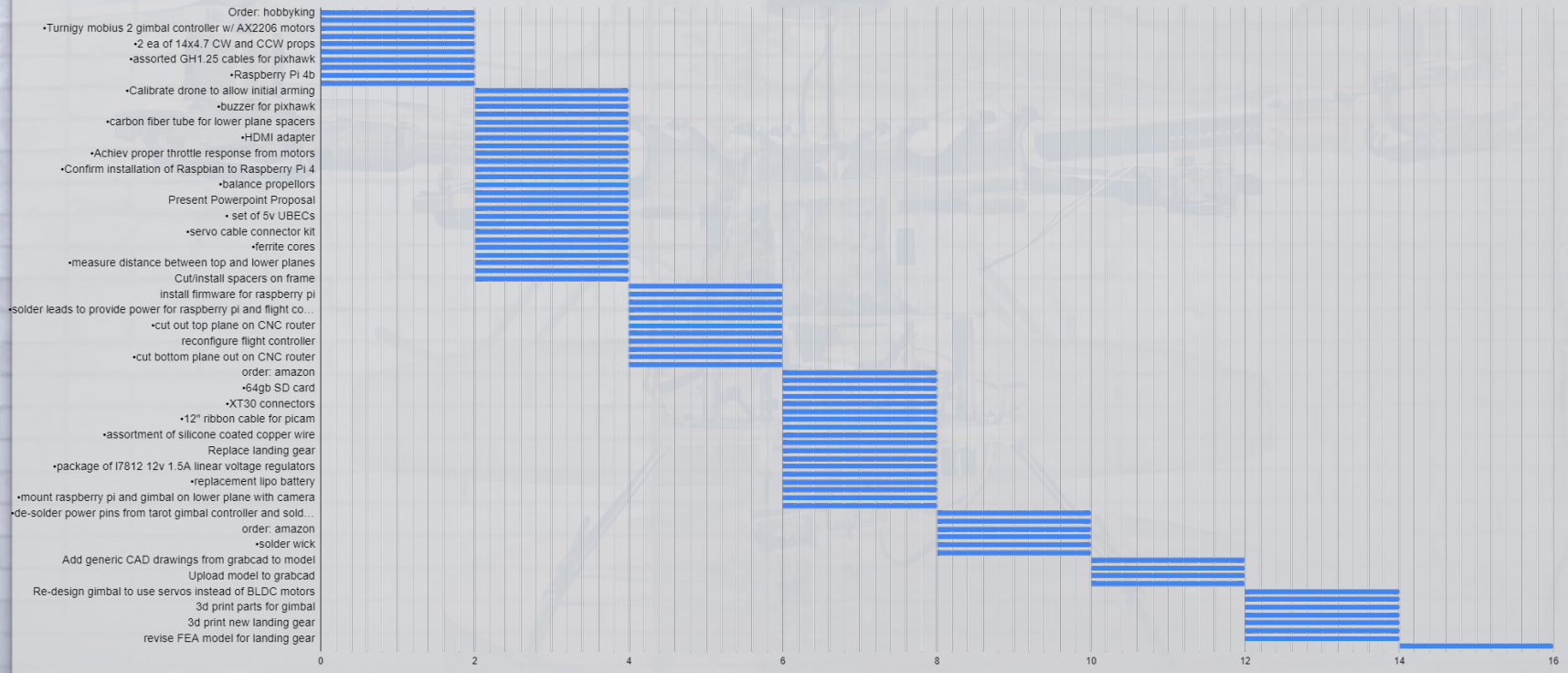
The following flight controllers were considered for this project and ranked in order to select the ideal one for this project

Flight Controller	Cost	Ease of Use	Support	Features	Total
Pixhawk Px4 V2	5	$4.00	$5.00	4	**18**
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

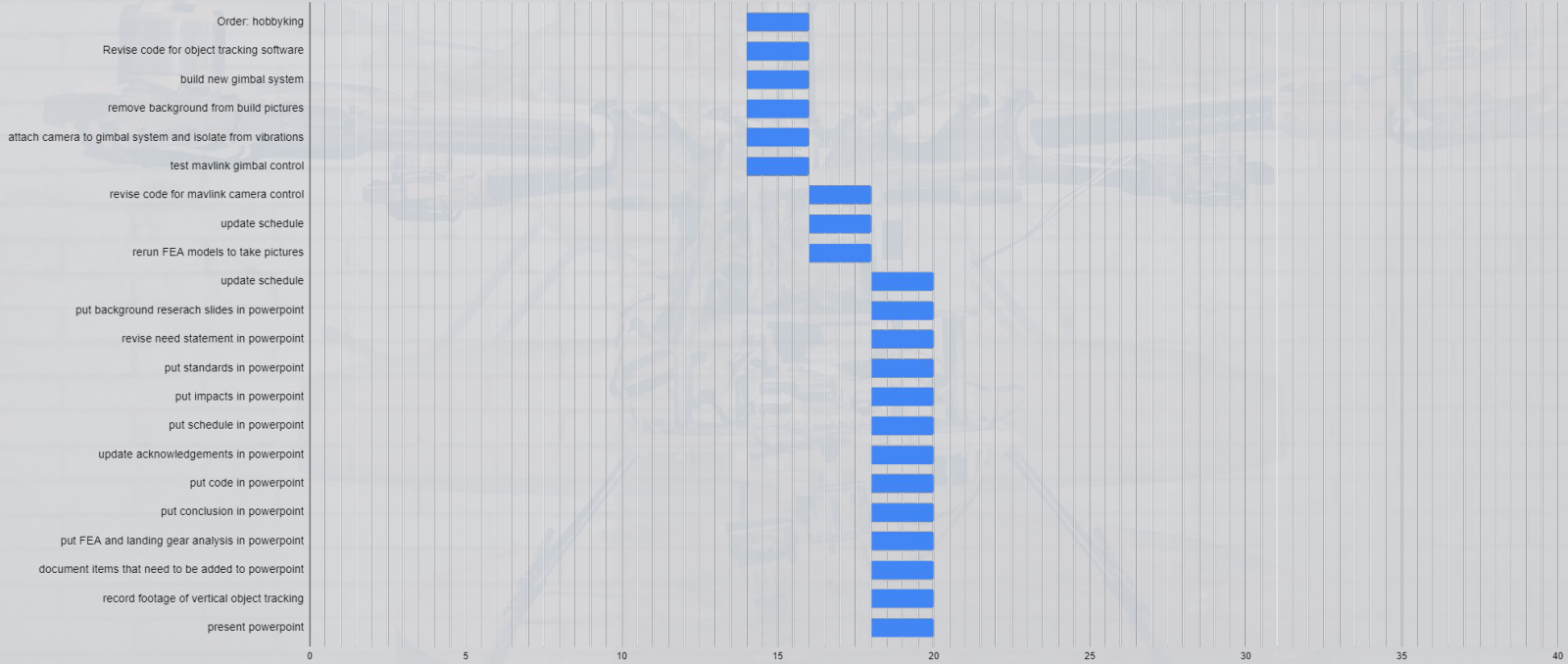
Development Cost Breakdown

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

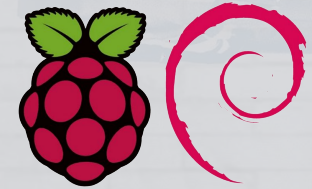

Order: hobbyking
•Turnigy mobius 2 gimbal controller w/ AX2206 motors
•2 ea of 14x4.7 CW and CCW props
•assorted GH1.25 cables for pixhawk
•Raspberry Pi 4b
•Calibrate drone to allow initial arming
•buzzer for pixhawk
•carbon fiber tube for lower plane spacers
•HDMI adapter
•Achiev proper throttle response from motors
•Confirm installation of Raspbian to Raspberry Pi 4
•balance propellors
Present Powerpoint Proposal
• set of 5v UBECs
•servo cable connector kit
•ferrite cores
•measure distance between top and lower planes
Cut/install spacers on frame
install firmware for raspberry pi
•solder leads to provide power for raspberry pi and flight co...
•cut out top plane on CNC router
reconfigure flight controller
•cut bottom plane out on CNC router
order: amazon
•64gb SD card
•XT30 connectors
•12" ribbon cable for picam
•assortment of silicone coated copper wire
Replace landing gear
•package of l7812 12v 1.5A linear voltage regulators
•replacement lipo battery
•mount raspberry pi and gimbal on lower plane with camera
•de-solder power pins from tarot gimbal controller and sold...
order: amazon
•solder wick
Add generic CAD drawings from grabcad to model
Upload model to grabcad
Re-design gimbal to use servos instead of BLDC motors
3d print parts for gimbal
3d print new landing gear
revise FEA model for landing gear
0
2
4
6
8
10
12
14
16

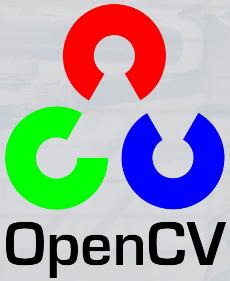
Scheduling


Order: hobbyking
Revise code for object tracking software
build new gimbal system
remove background from build pictures
attach camera to gimbal system and isolate from vibrations
test mavlink gimbal control
revise code for mavlink camera control
update schedule
rerun FEA models to take pictures
update schedule
put background reserach slides in powerpoint
revise need statement in powerpoint
put standards in powerpoint
put impacts in powerpoint
put schedule in powerpoint
update acknowledgements in powerpoint
put code in powerpoint
put conclusion in powerpoint
put FEA and landing gear analysis in powerpoint
document items that need to be added to powerpoint
record footage of vertical object tracking
present powerpoint
0
5
10
15
20
25
30
35
40

Software Stack:

Computer Vision Package



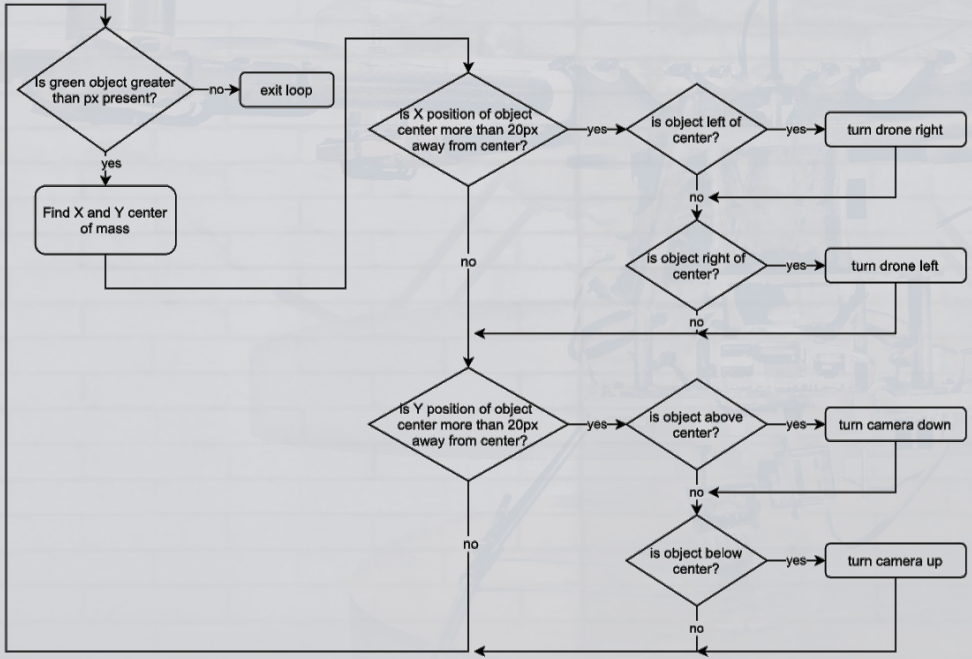


Configuration/Communication

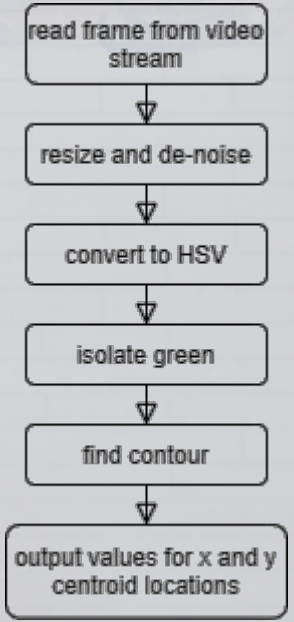
Flight Controller Firmware

Object Tracking:

Control Logic



OpenCV



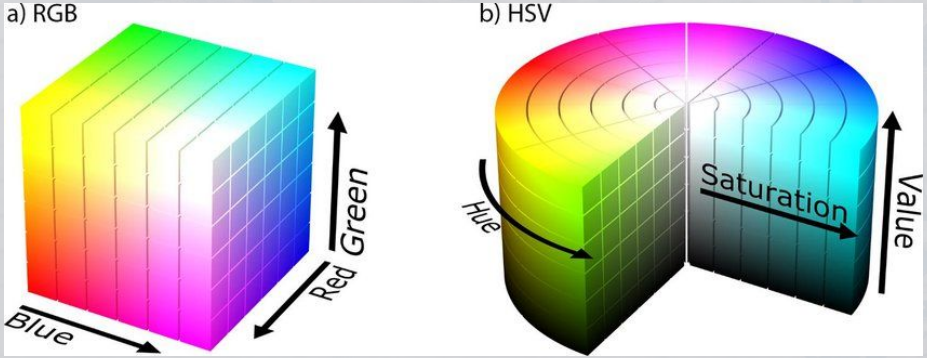
Code:

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# import functions for opencv
from __future__ import print_function
from collections import deque
from imutils.video import VideoStream
import numpy as np
import cv2
import imutils
import time
from dronekit import connect, VehicleMode, LocationGlobal, LocationGlobalRelative
from pymavlink import mavutil  # Needed for command message definitions
import math

# variables for camera/gimbal settings
x_max = 600  # width of pi camera image
y_max = 450   # height of pi camera image
x_center = x_max / 2  # center value used for readability/ease of editing
y_center = y_max / 2  # center value used for readability/ease of editing
gimbal_y_pos = -45 # goes from -90 to 0
pix_per_deg = 40 # used to adjust movement sensitivity

def mapObjectPosition(x, y):
    # print object coordinates
    print("[INFO] Object Center coordinates at X0 = {0} and Y0 =  {1}".format(x, y))

greenLower = (29, 86, 6) # define lower boundaries of green
greenUpper = (64, 255, 255) # define upper boundaries of green
pts = deque(maxlen=64) # list of tracked points

# load the webcam video stream
vs = VideoStream(usePiCamera=True).start()

# serial connection for pixhawk
connection_string = '/dev/serial/by-id/usb-ArduPilot_fmuv2_1C0034000951353332373834-if00'
print('Connecting to vehicle on: %s' % connection_string)
vehicle = connect(connection_string, wait_ready=True) # Connect to the Vehicle
vehicle.flush() # send a "clear all" type message

```

```python
# keep looping
while True:
    # grab the current frame
    frame = vs.read()

    # handle the frame from VideoCapture or VideoStream
    frame = frame[1] if args.get("video", False) else frame

    # resize the frame
    frame = imutils.resize(frame, width=600)
    # blur it
    blurred = cv2.GaussianBlur(frame, (11, 11), 0)
    # convert to HSV color space
    hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)

    # construct a mask for the color "green"
    mask = cv2.inRange(hsv, greenLower, greenUpper)
    # filter false positives
    mask = cv2.erode(mask, None, iterations=2)
    #filter false negatives
    mask = cv2.dilate(mask, None, iterations=2)

    # find contours in the mask and initialize the current
    # (x, y) center of the ball
    cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)
    center = None

    # only proceed if at least one contour was found
    if len(cnts) > 0:
        # find the largest contour in the mask
        c = max(cnts, key=cv2.contourArea)
        # compute the minimum enclosing circle
        ((x, y), radius) = cv2.minEnclosingCircle(c)
        M = cv2.moments(c) # find centroid
        center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))

        # only proceed if the radius meets a minimum size
        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

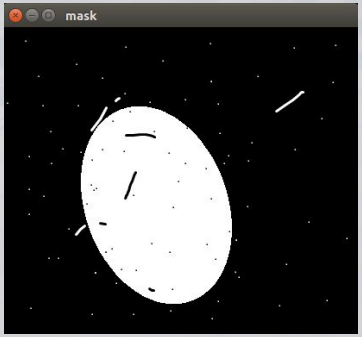
Image Processing:

Gaussian Blur

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

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

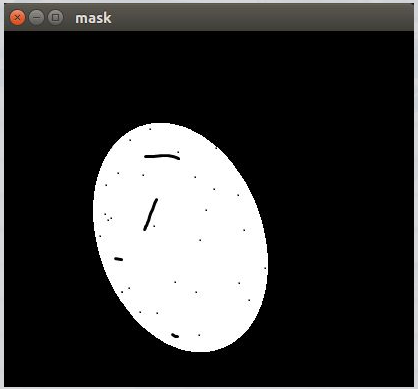
Mask



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

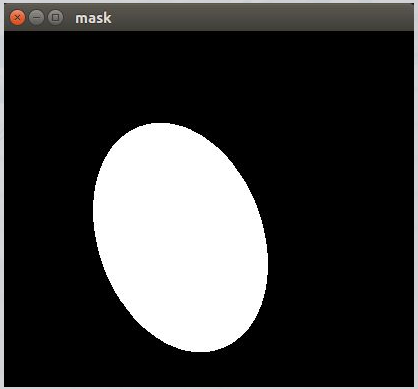
Image Processing:

Erode Filter



Removes false positives due to camera noise and other artifacts

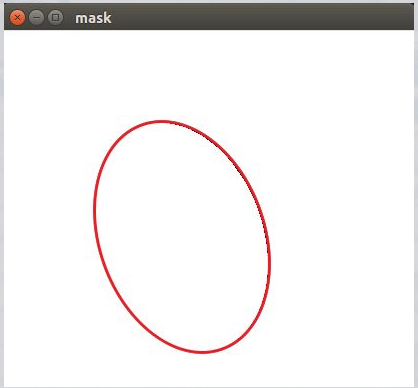
Dilate Filter



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

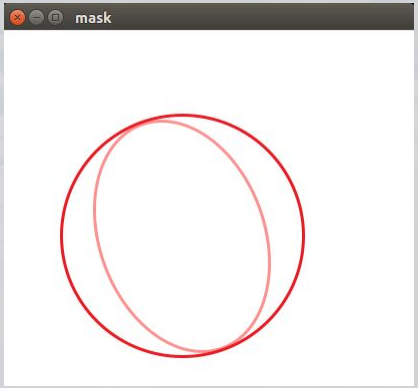
Image Processing:

Grab Contour



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

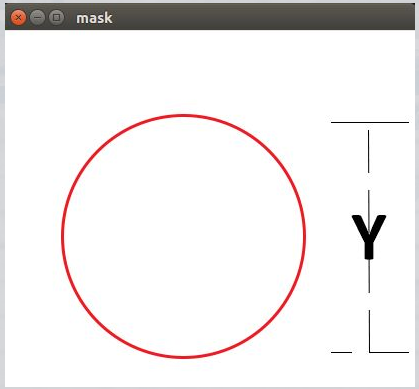
Find Enclosing Circle

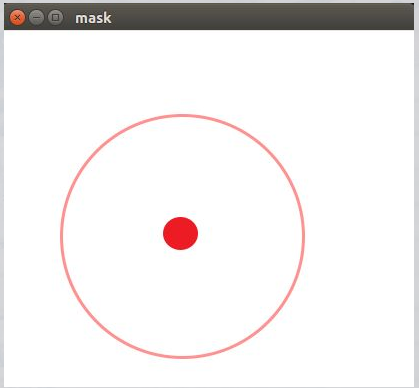


Finds the smallest circle which encloses the contour

Image Processing:

Find Centroid





Finds centroid based on mean area distribution in the enclosed circle

Code:

```
            # draw the circle
            cv2.circle(frame, (int(x), int(y)), int(radius),
                (0, 255, 255), 2)
            # draw the centroid
            cv2.circle(frame, center, 5, (0, 0, 255), -1)
            # update the list of tracked points
            mapObjectPosition(int(x), int(y))

            x = x_max - x # flip so zero = left
            y = y_max - y # flip so zero = bottom

            # adjust gimbal to follow y centroid
            if (y > (y_center + pix_per_deg)):
                print("above")
                gimbal_y_pos = gimbal_y_pos + (y - y_center)/pix_per_deg
            elif (y < (y_center - pix_per_deg)):
                print("below")
                gimbal_y_pos = gimbal_y_pos - (y_center - y)/pix_per_deg
            if gimbal_y_pos > 0:
                gimbal_y_pos = 0
            elif gimbal_y_pos < -90:
                gimbal_y_pos = -90
            vehicle.gimbal.rotate(int(gimbal_y_pos), 0, 0)

    # update the points queue
    pts.appendleft(center)

    # loop over the set of tracked points
    for i in range(1, len(pts)):
        # ignore non-valid points
        if pts[i - 1] is None or pts[i] is None:
            continue

        # Compute thickness of line
        thickness = int(np.sqrt(64 / float(i + 1)) * 2.5)
        # draw the connecting lines
        cv2.line(frame, pts[i - 1], pts[i], (0, 0, 255), thickness)

    # show the frame to our screen
    cv2.imshow("Frame", frame)
```

```
    key = cv2.waitKey(1) & 0xFF

    # if the 'q' key is pressed, stop the loop
    if key == ord("q"):
        break

# Close vehicle object
print("Close vehicle object")
vehicle.close()

# stop the camera video stream
vs.stop()

print("Completed")
```

Image Processing:

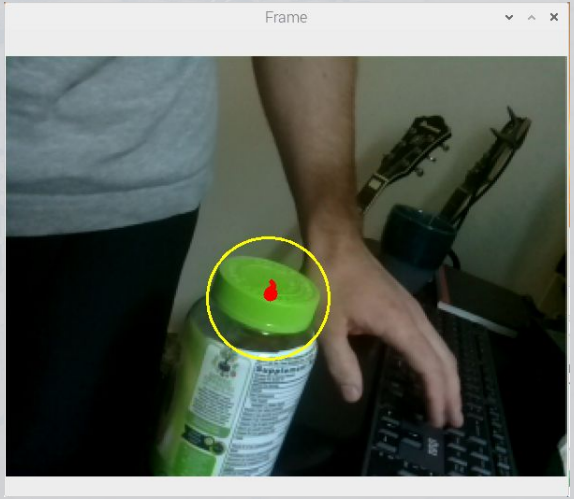
Centroid values output

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
- |Centroid – y center| > 40:
  - Rotate drone |Centroid – 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

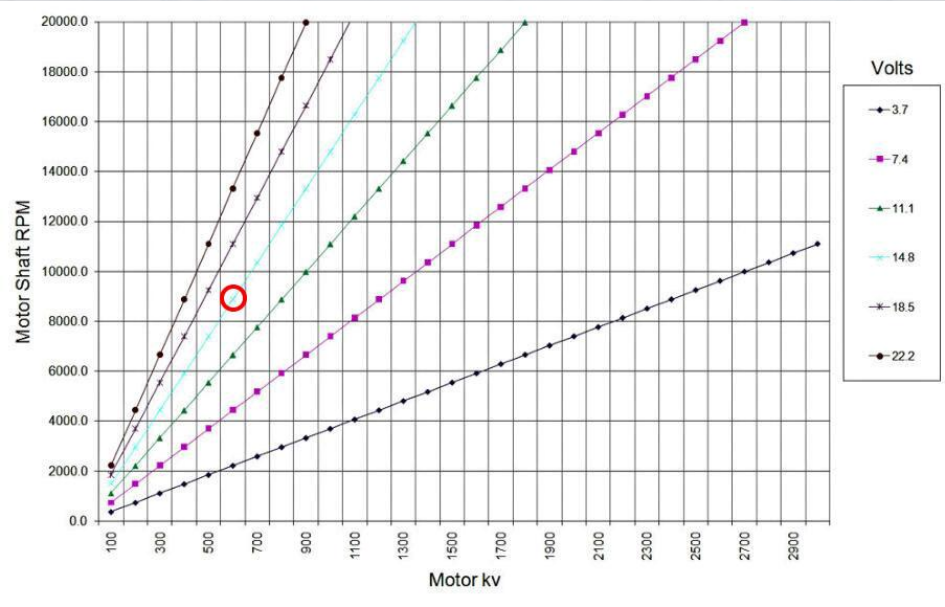
Object Tracking:

Yaw Trim Sensitivity

- 40px/deg value for close-range
- |Centroid – y center| > 40:
  - Move camera |Centroid – 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

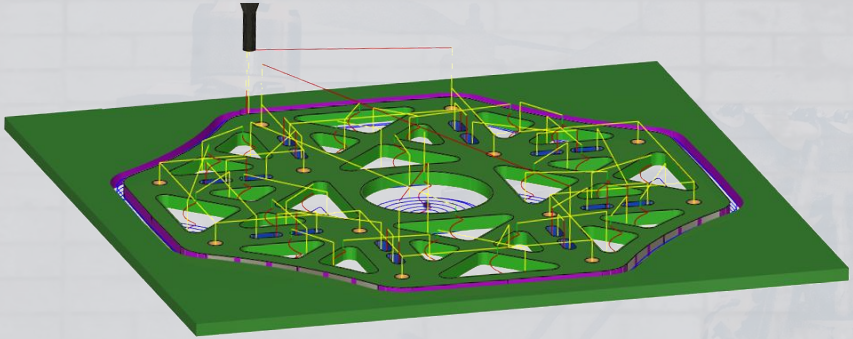
1. $UAV\ Weight = 6.4lbs$
2. $Static\ Thrust(N)$:

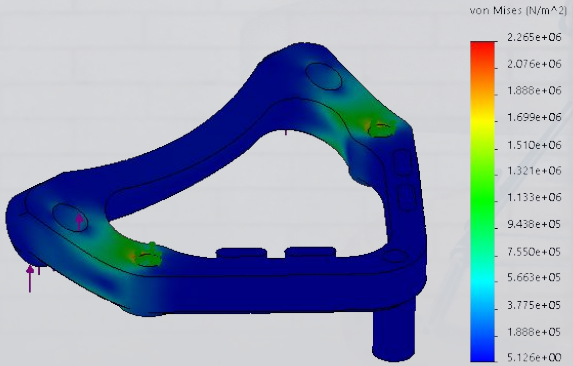
   $$1.86 * 10^{-11}\left(RPM^2 * \sqrt{Pitch} * D^{3.5}\right)$$

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. $Thrust\ to\ Weight\ Ratio = 1.13$

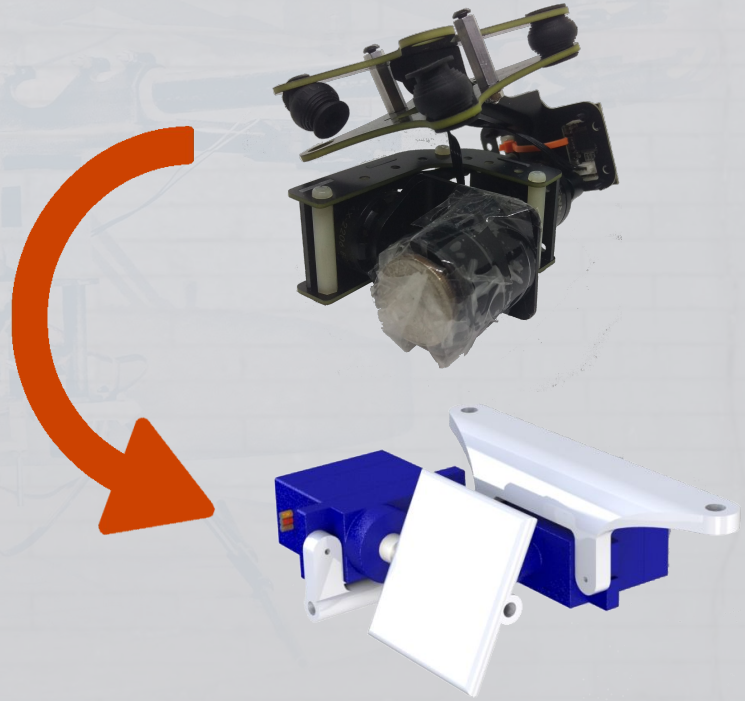
Development Lessons Learned:

Simulate





Simplify!

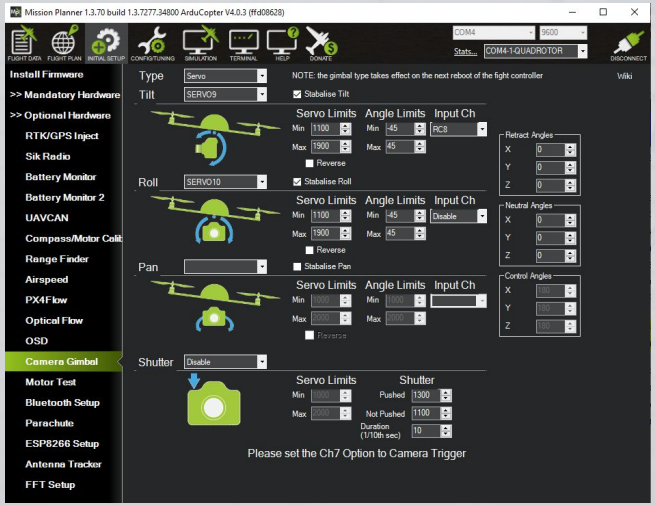


Development Lessons Learned:

Original Gimbal:

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

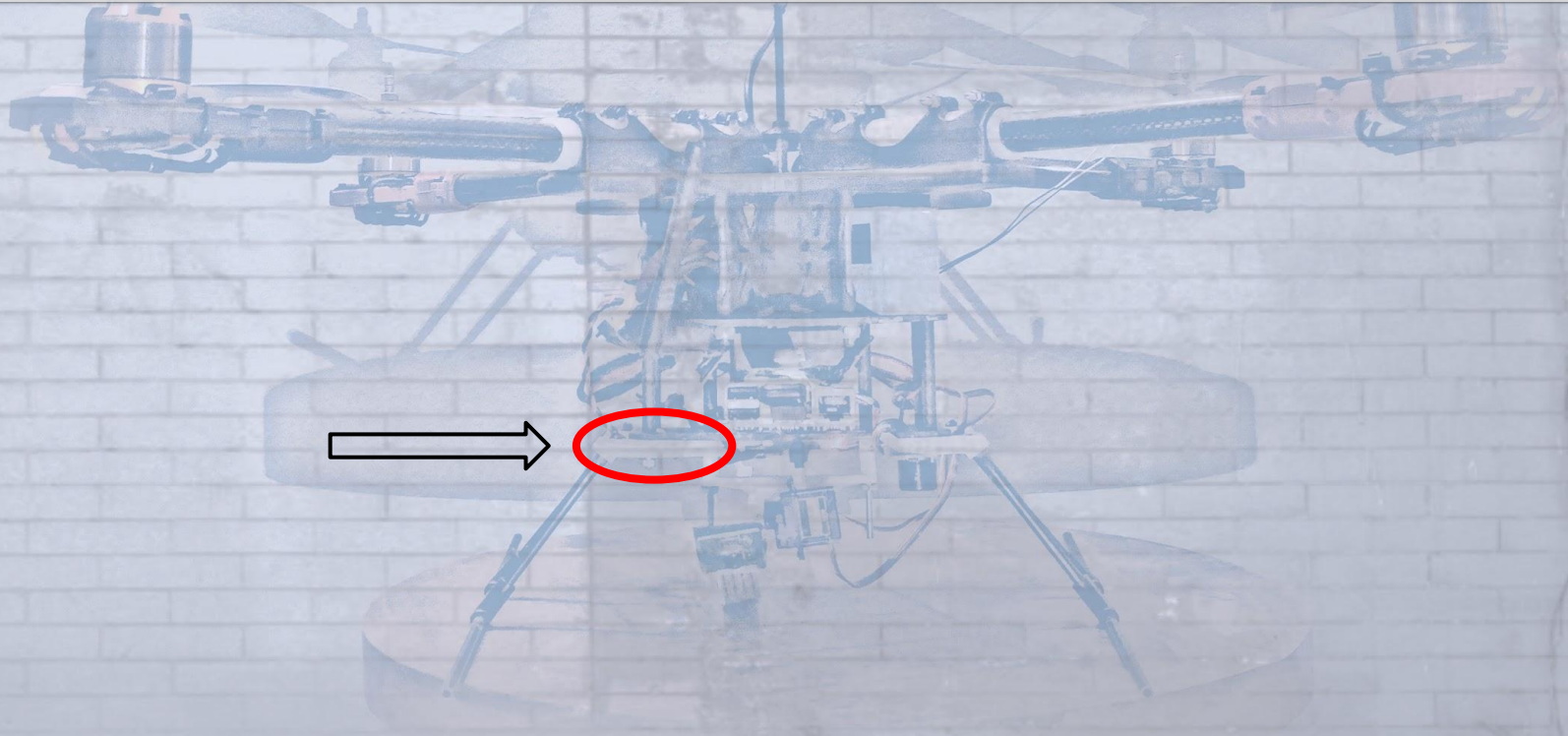
Development Lessons Learned:



Updated Gimbal:

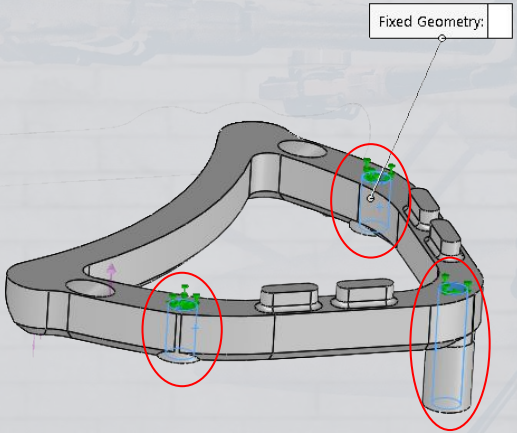
- Light
- Driven by flight controller
- Configurable through flight controller software

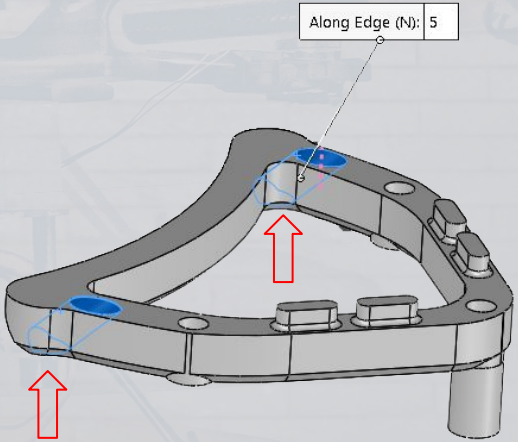
Designing the Landing Gear... for the Drone... for the Object Tracking System:



Methodology:

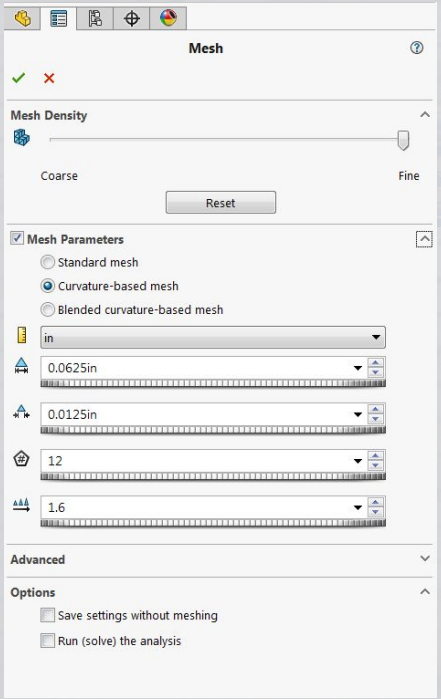
Iterative Design Process, Isolate 1 Variable to Change

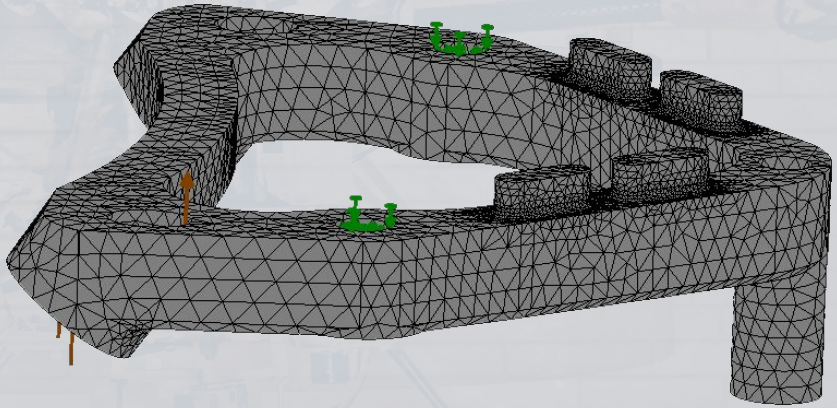




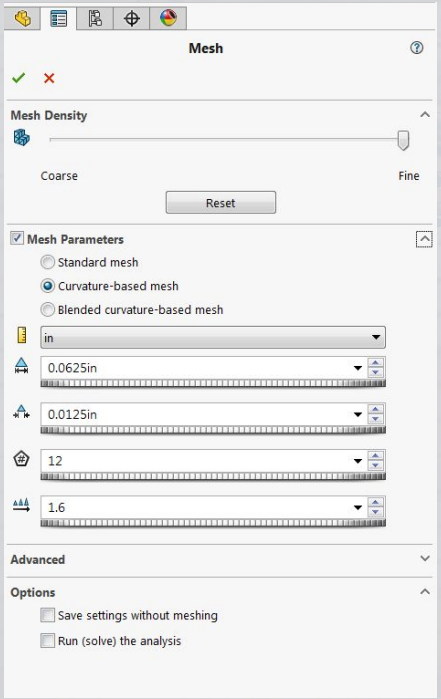
Consistent Loading across iterations

Creating a Representative Mesh Model





Creating a Representative Mesh Model



Maximum Tetrahedron Size: 1/16”

Minimum Tetrahedron Size: 1/80”

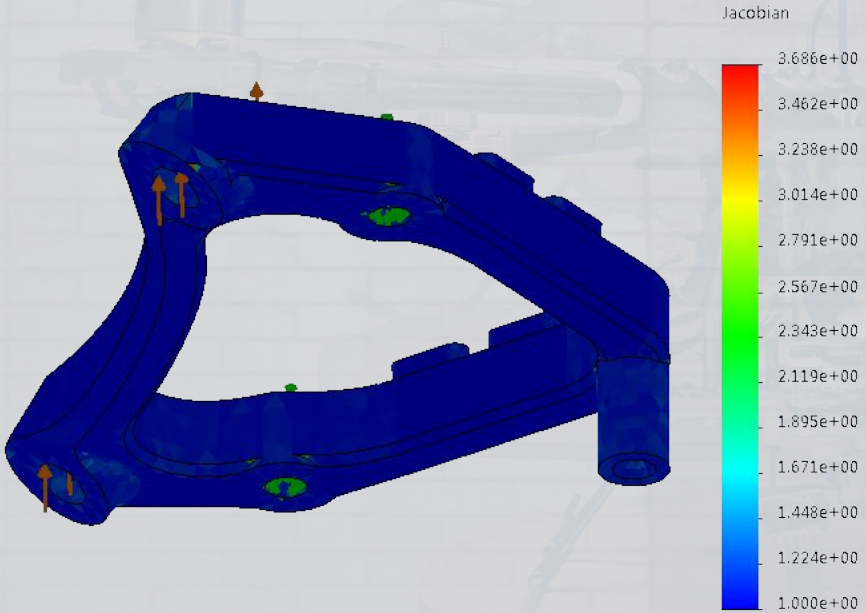
Minimum Edges/Circle: 12

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

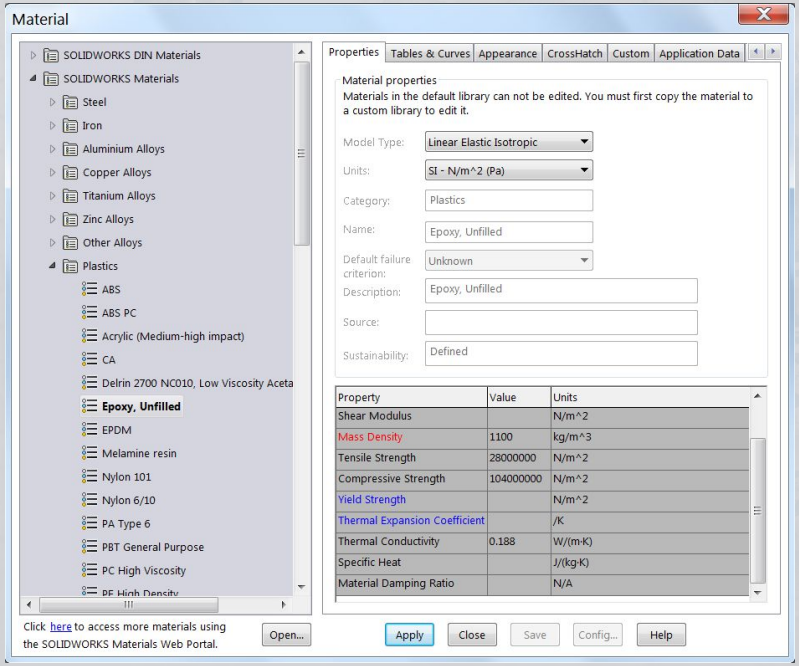
Force per leg = (9.81m/s^2 x 2kg)/4 ~=5N

Marin Factors: Not feasible to Use

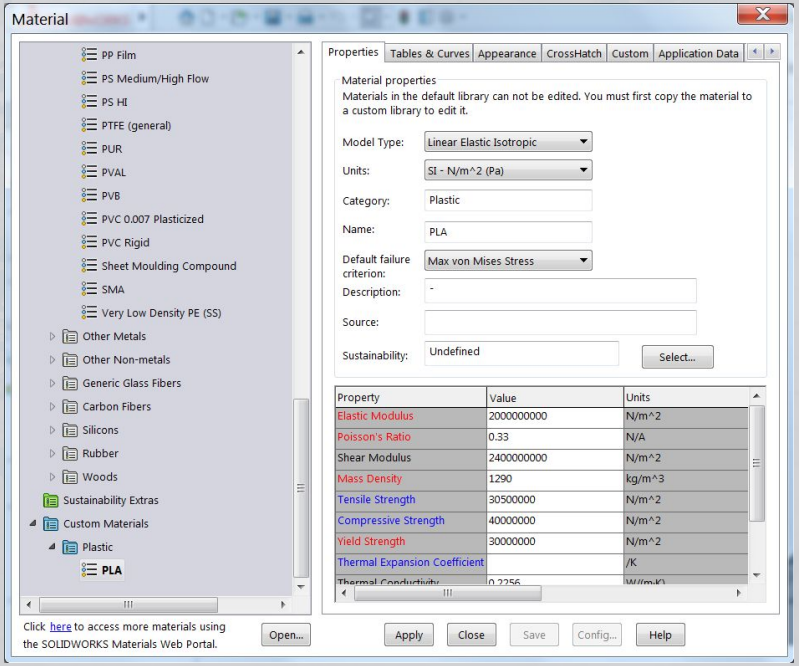
- Ka (surface factor) - Not available for PLA or Epoxy Resin
- Kb (size factor) - No model available for given cross section
- Kc (loading factor) = 1 (in bending)
- Kd (temperature factor) - Varies widely for PLA and Epoxy

Material Properties:

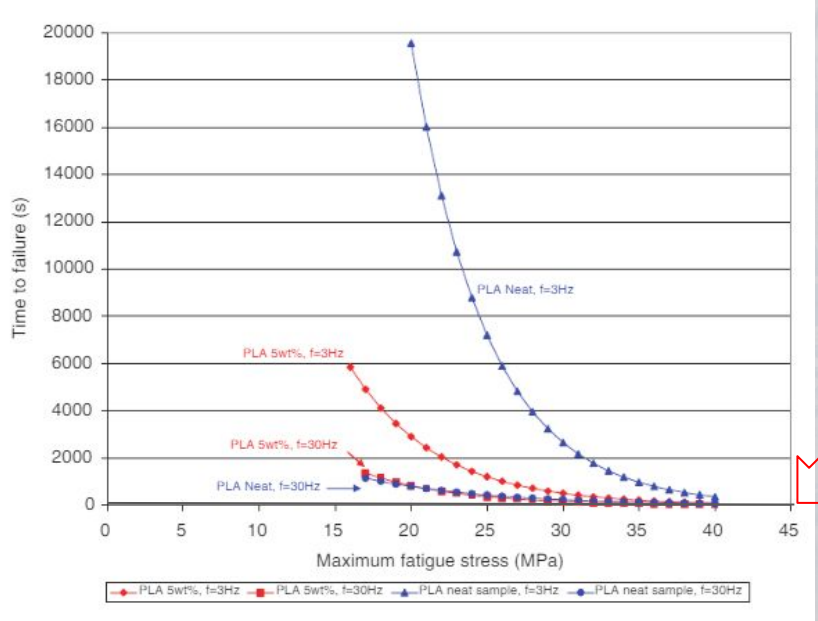
Epoxy



Polylactic Acid (PLA)



Endurance Limit:



*Mechanical Behaviour of Poly(Lactic Acid)

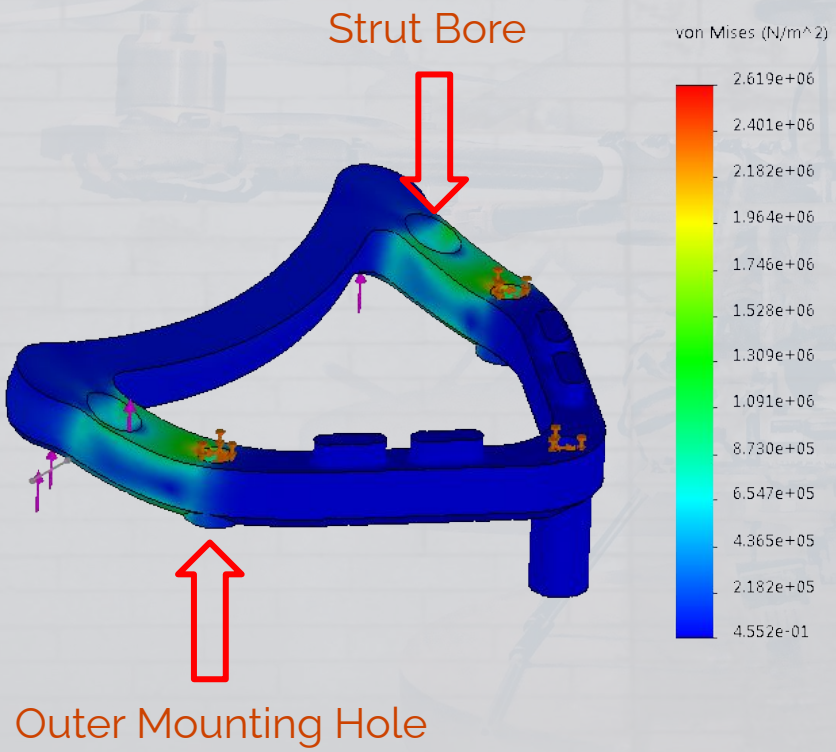
For less than 100 expected cycles:

100 cycles x 3hz = 33s

3s

Endurance limit is not relevant!

1st Iteration: Epoxy



Epoxy yield strength:

6.5e7N/m^2

Factor of Safety:

F.O.S = (6.5e7N/m^2)/(2.6e6N/m^2)

F.O.S = 25

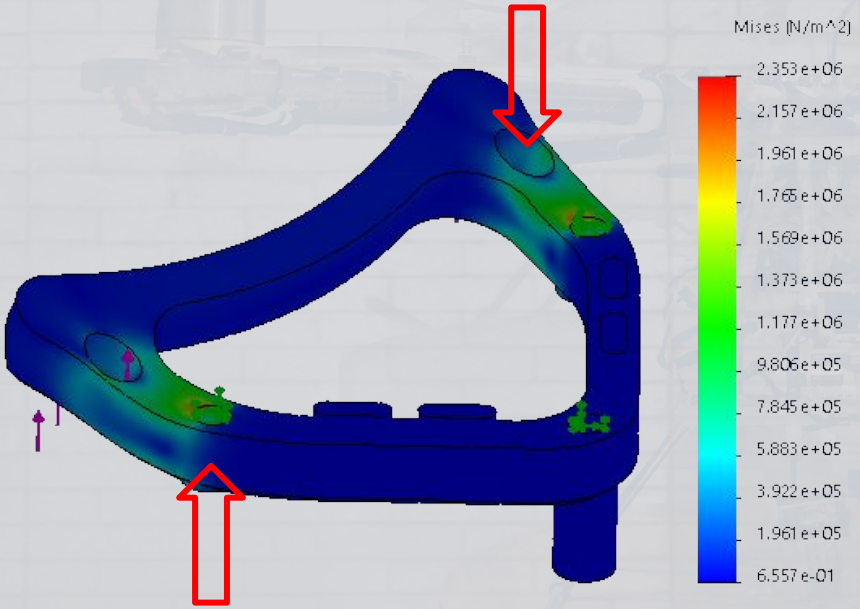
Location of Highest Stress:

Outer Mounting Holes

Failure Point:

Strut Bore Wall

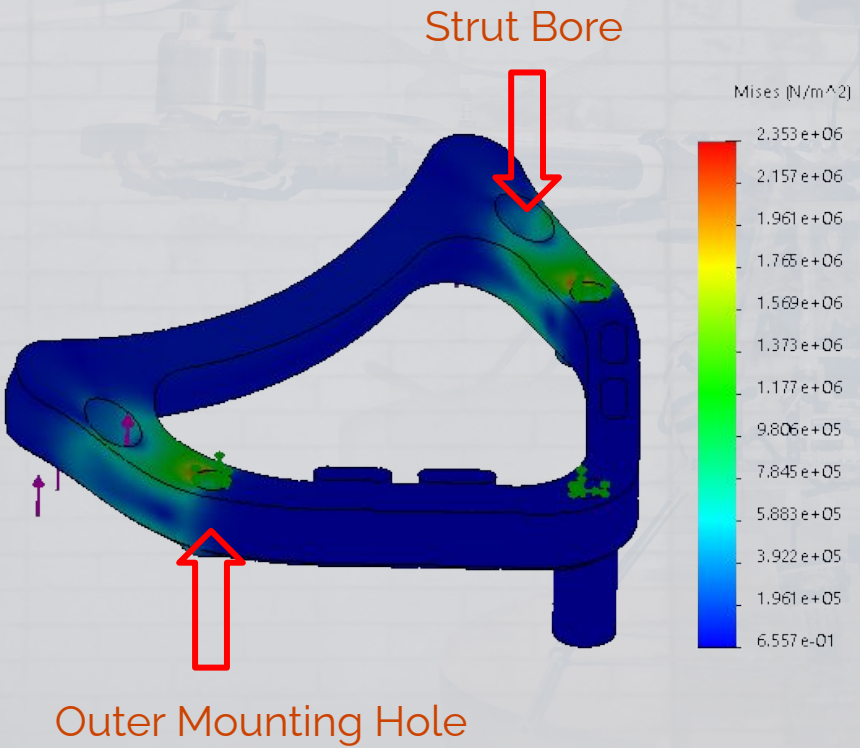
2nd Iteration: Epoxy



Changes Made:

Material added around outer mounting holes and strut bores

2nd Iteration: Epoxy



Epoxy yield strength:

6.5e7N/m^2

Factor of Safety:

F.O.S = (6.5e7N/m^2)/(2.4e6N/m^2)

F.O.S = 27

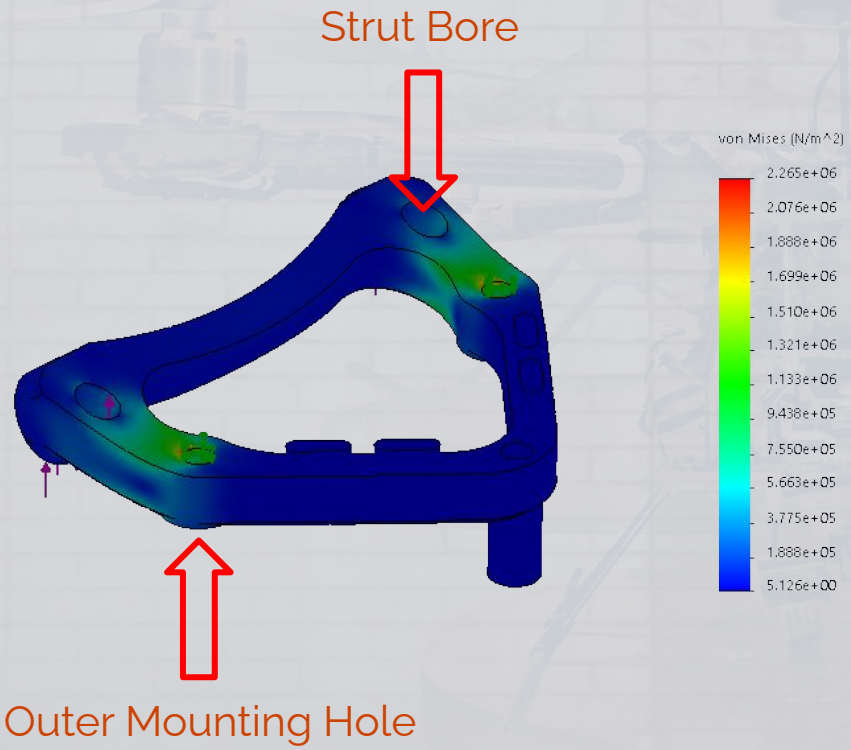
Location of Highest Stress:

Outer Mounting Holes

Failure Point:

Outer Mounting Holes

3rd Iteration: Epoxy

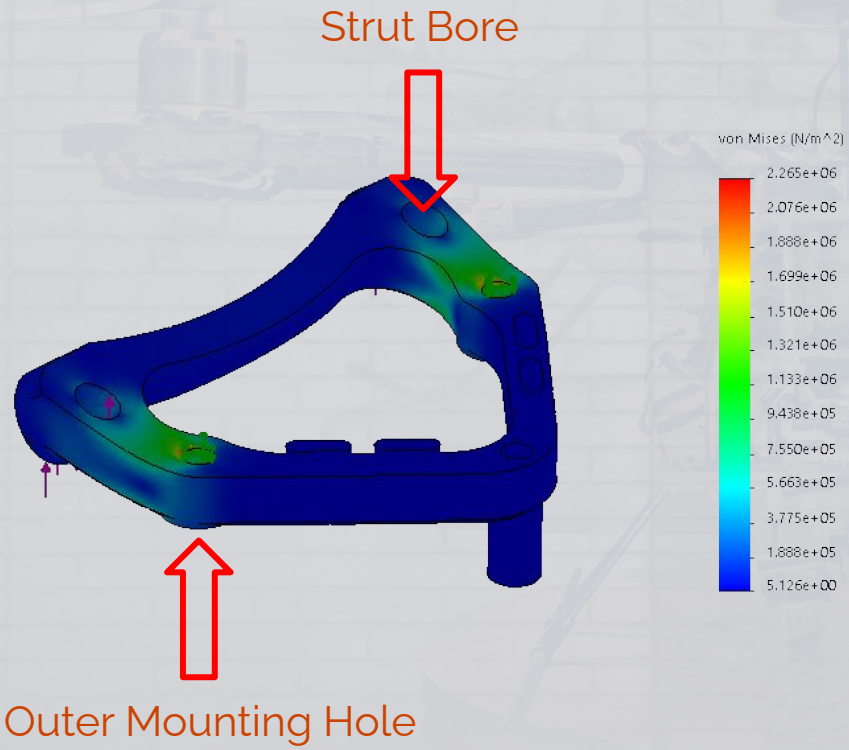


Changes Made:

Distance between strut bore and outer mounting hole decreased

Material added around mounting hole

3rd Iteration: Epoxy



Epoxy yield strength:

6.5e7N/m^2

Factor of Safety:

F.O.S = (6.5e7N/m^2)/(2.3e6N/m^2)

F.O.S = 27

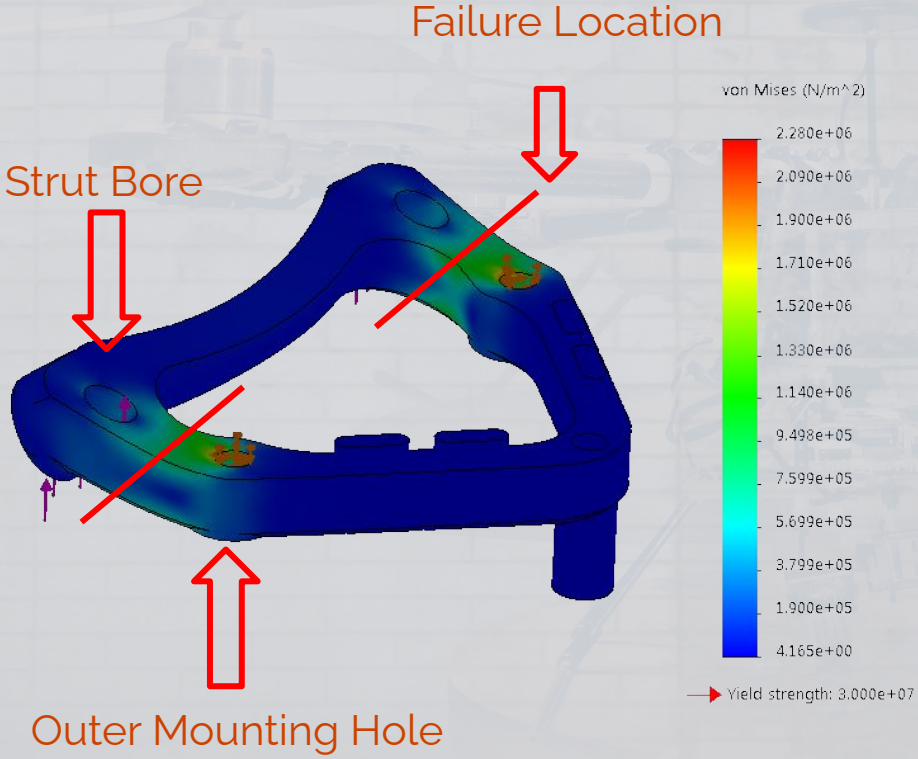
Location of Highest Stress:

Outer Mounting Holes

Failure Point:

Strut Bore Wall

3rd Iteration: PLA

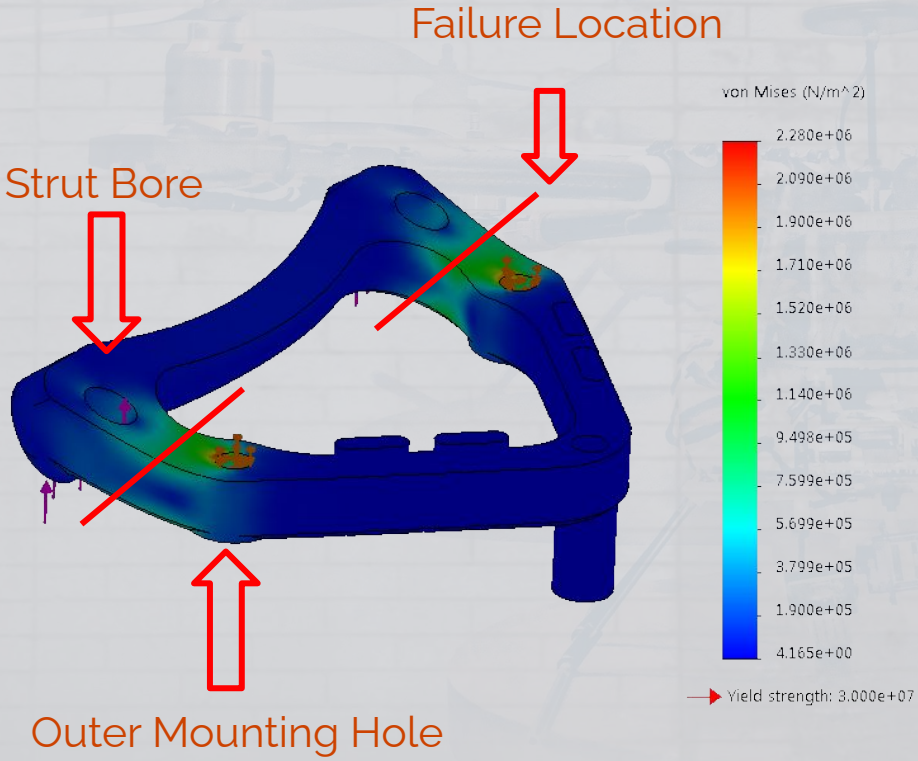


Changes Made:

Material changed to PLA

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

3rd Iteration: PLA



PLA yield strength:

5.6e7N/m^2

Factor of Safety:

F.O.S = (5.6e7N/m^2)/(2.2e6N/m^2)

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 ½ the force of the resin bracket even though it is stronger. One must consider real-world factors in design. While glass (yield strength = 7e9 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

$(\sigma_{Yeild})/(\sigma_{Max(static)})$ 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%.

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

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

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

29% weight reduction allows for lower load on:

- Propellers
- Motors
- ESCs
- Battery
- Frame

Will allow for:

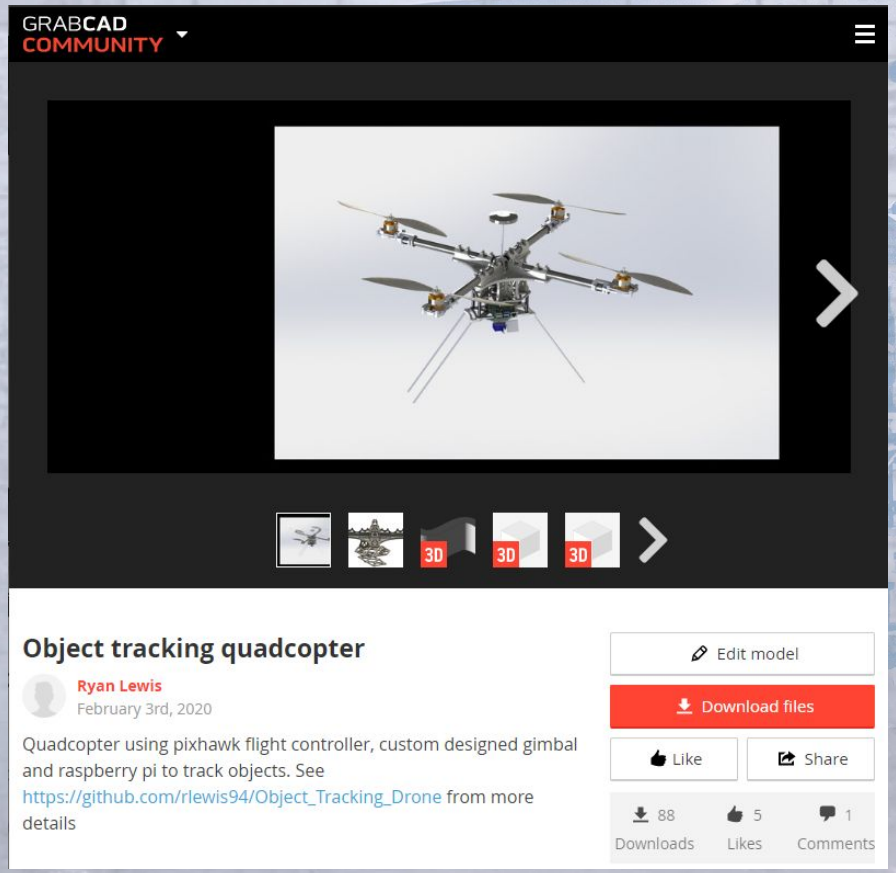
- Increased maneuverability in transient flight conditions
- Increased flight times

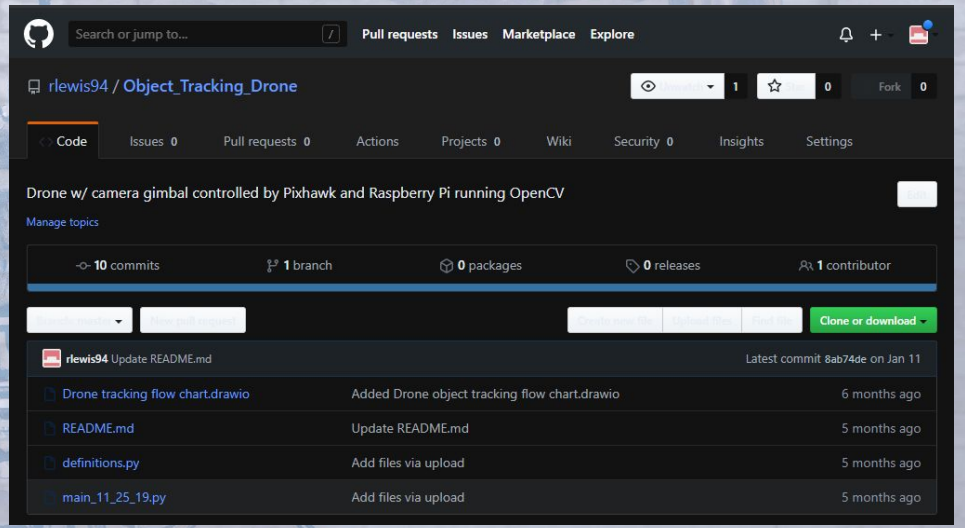
Weight:Thrust Reduction:

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:





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
- HSL and HSV. (2020, April 19). Retrieved from 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.