SUAS Competition - Software Team

FALL 2023 Final Report

Authors: Marc Cruz, Abdul Kalam Syed, Max Gross, Joshua Estrada, Jason Mar, Josh Ng, Ethan Tarrer, Sarkis Gafayan, Rubayet Mujahid, David Jackson

Status: Done

Date: Sep 21, 2023

Relative Links:

- SUAS Competition - Software Team Overview
- SUAS Competition - Technical Design Document
- SUAS Competition - Machine Learning Models

1. Introduction

This is a progress report for the Software team for SUAS 2024. This document contains current progress of the project, and also adding any new revisions that the SUAS 2025 Software Team to consider to increase the odds of the CPP SUAS team to win more consistently.

SUAS, or Student Unmanned Aerial Systems, is a yearly competition with a particular mission. For 2024 Competition, the mission is “Multiple package delivery companies have tasked UAS to deliver packages to customers. These UAS must avoid each other, travel to the customer, identify potential drop locations, and deliver the package to a safe location.” The competition is broken down into four mission tasks: Autonomous Flight, Obstacle Avoidance, Object Detection, Classification, Localization (ODLC), and Air Delivery. This report focuses on the progress on the implementation of the ODLC and Obstacle Avoidance mission tasks. For details of actual methodology please refer to SUAS Competition - Technical Design Document

For ODLC, we are required to detect and classify two types of objects: standardized objects & emergent objects. Standardized objects are 8.5” x 11” in size of various shapes, alphanumeric/s, color of shape, and color of alphanumeric.

Figure 1, Standard object (left; white A on blue triangle) and Emergent object (right; manikin dressed in clothes

The potential shapes according to the competition rulebook: circle, semicircle, quarter circle, triangle, rectangle, pentagon, star, and cross

The potential colors according to the competition rulebook: white, black, red, blue, green, purple, brown, and orange

These targets are to be detected by the minimum altitude of 75 ft, and ideally within the range of 85 - 90 ft. Then with the classification of the target to be able to determine localization and signaling payload drop and coordinates to UAV.

For Obstacle Avoidance, creating an effective range around the UAV to ensure safety of the drone. Would be creating the integration and system with a LiDAR and on-board flight computer to detect potential obstacles and to avoid obstacles accordingly.

Software Team Members:

- Abdul Kalam Syed
- Joshua Estrada
- Josh Ng
- Jason Mar
- Max Gross
- Ethan Tarrer
- Sarkis Gafafyan
- Marc Cruz

2. FALL 2023 Progress

For the SUAS project, there was more consideration of recruitment, teaching, and actual implementation towards SUAS mission tasks. With that in mind, all the information was written into a general timeline with inclusion of methodology designs and trade studies.

2.1 Timeline

Recruitment (August - September)

- Creating the UAV Lab interview system
- Advertising and interviewing members to join the project

Learning Phase (August - October)

- Introduce machine learning basics
- Cover different types of machine learning models
- Ensure all team members have a solid understanding

Data Collection and Dataset Phase (October - December)

- Highlight the importance of data collection and preparation.
- Create a dataset tailored to SUAS competition objectives.

Modeling Phase (October - February)

- Explain the process of model selection and experimentation.
- Encourage team members to explore pre-built models and consider custom CNN development.

Real-time Testing Phase (February - August)

- Describe the implementation process for the selected model on the UAV.
- Emphasize real-time testing and fine-tuning.

2.2 Recruitment (August - September)

During this phase, focused on overall SUAS recruitment for each subteam, including Autonomous flight team, Payload Team, UAV Design Team, Systems team and Software Team. During this phase, Marc Cruz created and oversaw the UAV Lab Interviews, where the SUAS project along with the other projects within UAV Lab, interviewed together to speed up the recruitment process. Consisted of 5 interviewers sorting through 40 applicants and passing the most optimal members derived from the secretary problem, or an adjusted optimal stopping theory. Geared, so that SUAS has individuals who are capable of finishing the mission tasks, or the core members. But a larger emphasis on individuals who are able to work on the project for extended time of 2-3 years to solidify a solid group for future SUAS competition. Then once a solid core group is created, consideration of recruitment of Sophomore or Junior standing individuals to learn and later on take over graduating core member positions. These individuals are considered the auxiliary members. [Marc Cruz] Although I want to win the competition during my time as software lead, I also wanted to ensure that the project has longevity in terms of the knowledge that is accumulated from each year's competition. Previous years lacked proper documentation or available members to transfer information to the current competition year. Thus, the strong emphasis on the recruitment phase as it will determine if SUAS and the other projects can make substantial progress that ultimately leads to the completion and final product of UAV lab projects. Except in terms of SUAS, faster completion of mission tasks and allow room for even more advanced implementation to be considered.

2.3 Learning Phase (August - October)

2.3.1 ODLC

As new members are introduced and onboarded, each member that was on-boarded lack a strong foundation in machine learning or data science. With that in mind, to mitigate this problem the learning phase was considered and implemented. With the help of David Jackson, the list of libraries to familiarize themselves with relative libraries, including NumPy, OpenCV, and PyTorch.

For NumPy, It would be good to cover:

- Basic operations,
- Mathematical functions
- Reshaping arrays
- Combining arrays
- Element wise operations
- Linear algebra – possibly optional, but good to review.
- array statistics - possibly optional, but good to review.

For OpenCV (Computer Vision Library used to clean the image up to pass through to the machine learning library).

- Video in and out

- Cutting/Slicing portions of images
- Binary Image processing
- thresholding
- erosion and dilation
- opening and closing
- Contours
- Blob Detection
- Connected Component Analysis (connected blobs)

For PyTorch – PyTesseract – Keras-OCR, machine learning, and pre-trained text recognition models

- Using pretrained models in pytorch.
- Using PyTesseract for character recognition.
- Using Keras-OCR for character recognition.
- Identifying which has better results for the application at hand.

2.3.2 Obstacle Avoidance

For obstacle avoidance, as the topic is new to software team, mimic the usage of other drones within the same lab. The implementations that were considered by other teams were the usage of LiDAR to detect obstacles and sending an output to their pixhawk, or whatever hardware managing the flight path, and rerouting away or around the obstacle. Looked into various open source LiDAR implementations, such as, MITs “mader” that is capable of generating initial flight path and creating a spline to be followed that goes around the object.

2.4 ODLC Design

During this phase, the methodology was further fleshed out and the general design of the multi modal machine learning application. An overview of the general, consists of multiple machine learning models stacked upon each other to consider multiple traits of the two types of targets ODLC is required to find. The illustration below, illustrates the methodology of how the machine inference system should operate.

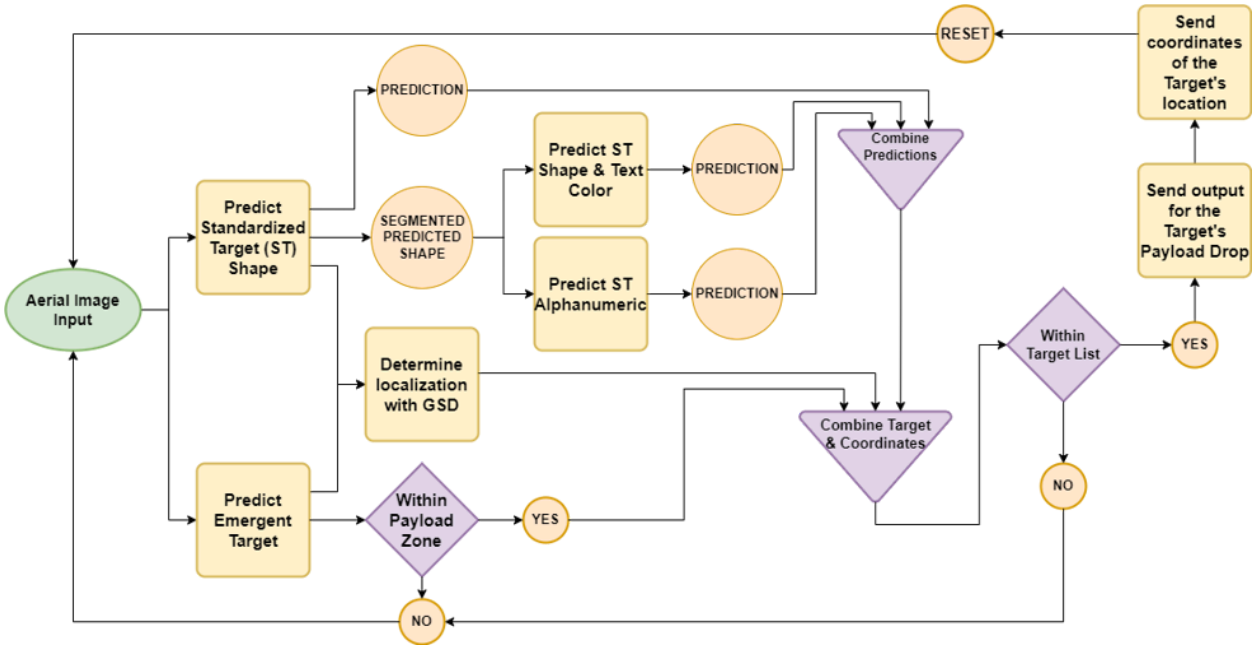

Figure 2 Flowchart of the machine inference methodology

The Machine Inferencing Methodology consists of 4-5 Machine Learning models that correspond to a trait: Predicting standardized object's shape, shape color, alphanumeric color, and alphanumeric. Chosen to combine multiple models instead of considerations of a cluster in order to utilize the current software team's knowledge. Then the illustration below depicts the hardware integration of the machine inference methodology.

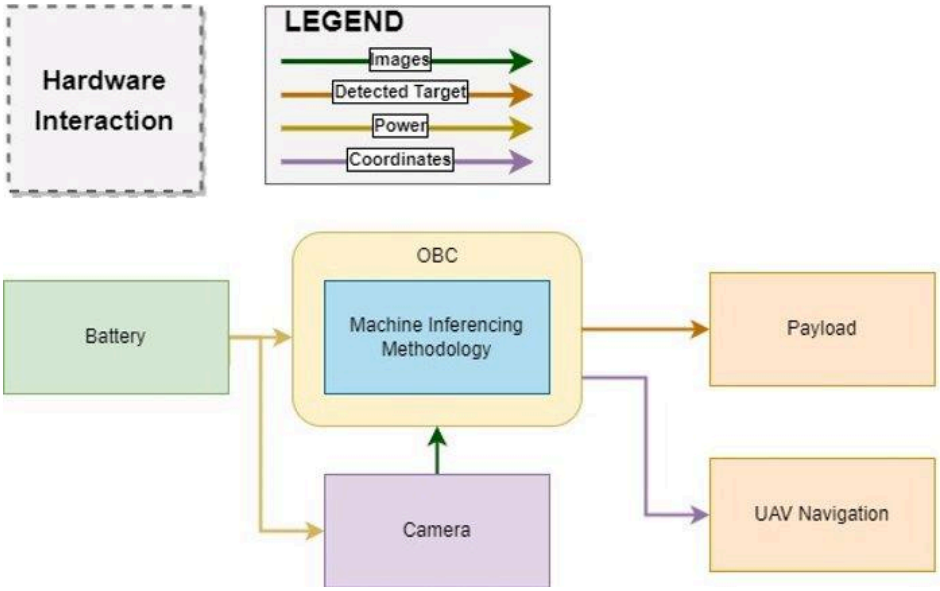

Figure 3 illustrates the hardware interactions between OBC, Camera, and other equipment on the UAV drone.

2.5 Hardware Trade Studies

This section contains the various trade studies that were conducted for required components to conduct ODLC and obstacle avoidance.

2.5.1 ODLC Trade Studies

2.5.1.1 OBC Trade Study

OBC Trade Study										
		Raspberry Pi 5 + Coral Accelerator			Raspberry Pi 4 + Coral Accelerator			Jetson Orin Nano		
Criteria	**Weight**	**Description**	**Grade**	**Weighted**	**Description**	**Grade**	**Weighted**	**Description**	**Grade**	**Weighted**
Cost	2	$171	3	6	$170	3	6	$500	1	2
Processing Units	5	Quad-core 64-bit Arm Cortex-A76 CPU	1	5	Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz	1	5	1024-core NVIDIA Ampere architecture GPU with 32 Tensor Cores	3.5	17.5
Software Support	1	Yes	3.5	3.5	Yes	3.5	3.5	Yes	3.5	3.5
Power Consumption (W)	3	12	2	6	8	3	9	10-22	2	6
Weight (g)	1	48	3.5	3.5	47	3.5	3.5	176	2	2
Average Grade				2.6			2.8			2.4
Weighted Total				24			27			31

OBC Normalization Chart				
Grade	**1**	**2**	**3**	**3.5**
Cost ($)	300+	< 300	<= 200	<= 100

Processing Units	Reduced Accuracy / Subpar Processing	Subpar Optimal	Semi Optimal Processing	Optimal Processing
Software Support	Software Obsolescence			Software is update to date
Power Consumption (W)	13+	<= 12	<= 8	<= 5
Weight (g)	200+	< 200	< 100	< 50

2.5.1.1 Camera Trade Study

Camera Trade Study										
		OAK-D Pro W PoE (IMX378 Sensor)			**SIYI ZR10**			**Rpi HQ Camera + 6mm Lens**		
Criteria	**Weight**	**Description**	**Grade**	**Weighted**	**Description**	**Grade**	**Weighted**	**Description**	**Grade**	**Weighted**
Cost ($)	1	599	1	1	506	1	1	75	3.5	3.5
Weight (g)	1	184	3	3	381	1	1	83.4	3.5	3.5
Resolution/FPS	5	1080P/60FPS 4K/30FPS	3.5	17.5	2560 x 1440p30	3.5	17.5	- 2028 × 1080p50 - 2028 × 1520p40 - 1332 × 990p120	3.5	17.5
Power Consumption (W)	3	2W - 5.5 W	3	9	3 W	3	9	1.4 W	3.5	10.5
FOV	4	HFOV: 95° VFOV: 70°	3.5	14	HFOVL 71.5° DFOV: 79.5 °	3	12	63°	3	12
Average Grade				2.8			2.3			3.4
Weighted Total				44.5			40.5			**47**

Camera Normalization Chart				
Grade	**1**	**2**	**3**	**3.5**
Cost ($)	400+	400 - 250	< 250 - 100	< 100
Weight (g)	350+	< 350 - 200	< 200 - 50	< 50
Resolution/FPS	Incapable of High Resolution @ 30 FPS			High Resolution @ 30 FPS
Power Consumption (W)	10+	< 10 - 5	< 5 - 2	< 2
FOV	Subpar optimal FOV		Moderately optimal FOV	Strongly optimal FOV

2.5.2 Obstacle Avoidance Trade Study

LiDAR Trade Study										
		RPLIDAR A2M8			**Livox Mid-40**			**RPLIDAR A3**		
Criteria	**Weight**	**Description**	**Grade**	**Weighted**	**Description**	**Grade**	**Weighted**	**Description**	**Grade**	**Weighted**
Cost ($)	1	319	3	3	599	3	3	599	3	3
Detection Range (m)	5	0.1 - 16	3.5	17.5	260	3.5	17.5	8	2	10
Software Support	3	Yes	3.5	10.5	Yes	3.5	10.5	Yes	3.5	10.5
Power Consumption (W)	4	3	3	12	10	1	4	3	3	12
Weight (g)	2	190	2	4	760	1	2	190	2	4
Average Grade				3			2.4			2.7
Weighted Total				47			37			39.5

LiDAR Normalization Chart				
Grade	**1**	**2**	**3**	**3.5**
Cost ($)	800+	<=800	<=400	<=200
Range (m)	<6	<= 8	<= 12	12+
Software Support	Software Obsolescence			Software is update to date
Power Consumption (W)	5+	4	3	<= 2
Weight (g)	200+	<200	<150	<100

2.6 Data Collection and Dataset Phase (October - December)

During this phase, members focused on working the ODLC mission task were split between all the models and focused on researching and collecting the required data for their specific machine learning model.

2.6.1 Standardized Object shape ML model

For the data collection for detecting shapes of the standardized object, the nature of the target's unique characteristics lead to the need of collecting its own dataset. Due to the difficulty of needing a drone pilot with licensing, and limited scheduling with current licensed drone pilots, led to consideration of synthetic dataset that combines aerial image datasets [1][2][3][4] and scripting potential variations of standardized targets onto each image. In figure 4, it depicts some of the aerial images from the dataset. Although these datasets are not the ideal conditions we would be competing in, due to the current circumstances the model will rely on the datasets that have over 100,000 images in just one of the several considered datasets. Then for the environment of the aerial images is still addresses competition environment as the competition environment is runway blacktop that has the same characteristics of the roads in the aerial image datasets.

In Figure 4, various samples from the aerial datasets

Starting with one of the datasets [3], began experimenting scripting standardized targets onto the images and made it so that the images were within a certain size percentage. Due to the nature of the dataset not having a consistent altitude, it was determined to create relative sized images of the standardized targets based on the rulebook. Several samples depicted below.

Figure 5, First created samples of experimental scripting of standardized objects onto aerial images.

To test the validity of the synthetic dataset collected, we trained YOLOv5 on the custom dataset. The model was only trained to detect circles in different colors with colored alphanumeric. The

model was trained on 750 training images plus 500 validation images (out of 24,000 images collected for the circle), and the results were promising. As shown in Figure 6, the trained model can detect very small circles with consistency. However, it is not very accurate as it recognizes any round object as a circle. The confusion matrix had a score of 100% which should not have been the case as it still doesn't detect circles of different sizes. The cause of the confusion matrix being so high is possibly overfitting, the training model and validation did not contain the same images but they were similar. Introducing more shapes should make the model more accurate as well.

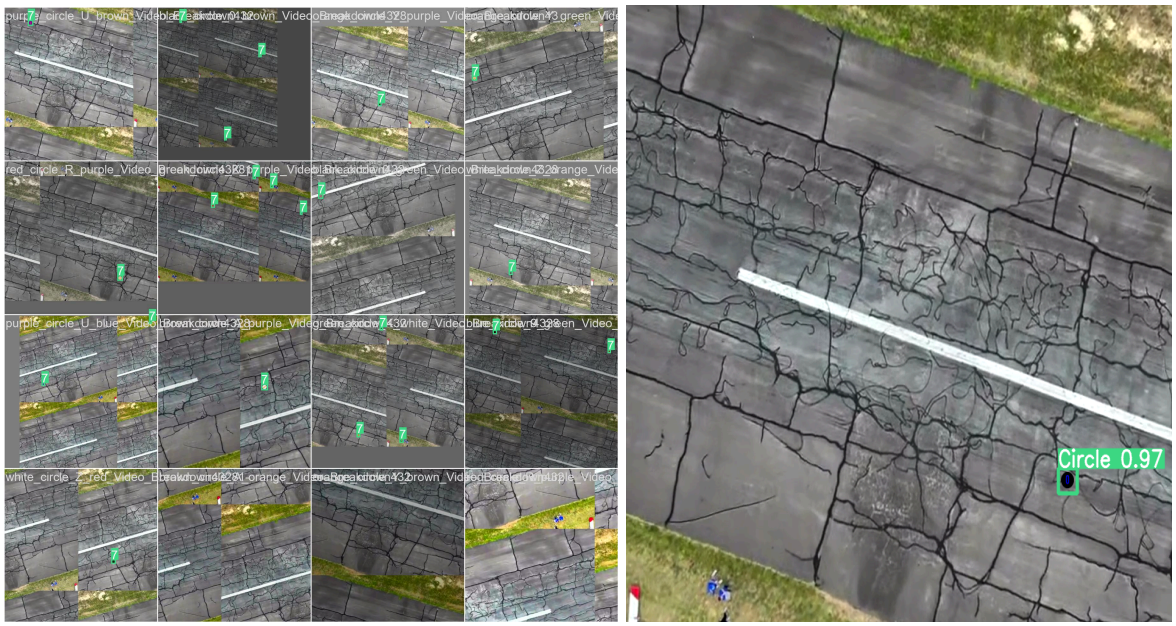

Figure 6, Trained YOLOv5 model results, being able to detect small colored circles with colored alphanumeric.

The scripting of images is currently experimental and more work would be completed during the winter break. Further work mentioned in Future Work section.

2.6.2 Standardized Object Shape & Alphanumeric Color ML model

Before creation of the Color ML model, focused on preprocessing images in order to increase the odds of predicting the color of the shape and alphanumeric colors. With the given list of colors from the rulebook, we are taking an image of a segmented predicted shape that is a result from the standardized object shape model. As the image being segmented, the images majority colors should be of the shape color and alphanumeric color. With that in mind, we extract RGB values of the shape and alphanumeric, or the 2 most common values in the image. This is the approach we chose to determine the RGB values of the image, then we move onto various methods of making a model. The two approaches are using the finite range RGB values (0-255) and make a range for each RGB value that determines if it is a certain color. For example, it is known that (0,0,0) is black or (255,255,255) is white. Then the second approach is creating or finding a RGB value dataset that has various RGB values that are labeled to its

corresponding color. Then by crossreferencing the pixel values to their rgb equivalent,the color that is most prevalent in the picture will be used to determine overall color. Using this same strategy, the color of the alphanumeric character can also be deduced by further shrinking the bounding box.

2.6.3 Standardized Object Alphanumeric ML model

With using openCV OCR to be able to conduct text detection that allows the alphanumeric to be read and output. Data collection was not required, but reimplementation of the openCV OCR was the main focus for this model as it is already prebuilt. To do so, using the same segmented image given to the color ML model, the image is taken as an input and outputs the text in the given image. This model acts as the main method of ensuring no false positives as occurences of a non-standardized target being a certain shape would not be considered a target if no text can be derived from the image with a high confidence threshold (90% confidence). The threshold may change depending on future experimentation in the spring semester.

2.7 Modeling Phase (October - February)

Due to the software team's lack of expertise in the subject of object detection models and the preprocessing of images related led to need to research information on implementation of various machine learning models.

2.7.1 UAV Image Recognition Trade Study / Decision Matrix

Methodologies Under Study:

- Convolutional Neural Networks (CNNs)
- Faster R-CNN
- YOLO (You Only Look Once)

Criteria for Evaluation:

- **Accuracy**: Ability to correctly identify and classify objects.
- **Speed**: Real-time processing capability.
- **Resource Requirement**: Computational and memory demand.
- **Training Data Dependency**: Amount of data required for effective training.
- **Implementation Complexity**: Difficulty in integrating with UAV systems.

Methodology Analysis:

1. Convolutional Neural Networks (CNNs)

- **Description**: Hierarchical deep learning models specialized for image and pattern recognition tasks.
- **Pros**:
    - **Layered Architecture**: Enables feature extraction at multiple levels, from simple to complex.

    - **Transfer Learning**: Pre-trained models available which can be fine-tuned for specific tasks.
    - **Adaptable**: Can be designed for varied complexities, from simple to very deep networks.
    - **Robustness**: Performs well even with minor image distortions or variations.
- **Cons**:
    - **Overfitting**: Prone to overfitting, especially with limited data.
    - **Computational Intensity**: Requires powerful GPUs for training and inference.
    - **Training Time**: Deep CNNs can take substantial time to train.
    - **Interpretability**: Difficult to interpret and understand layer-wise operations.
- **Research References**: [5], [6]

2. Faster R-CNN

- **Description**: Advanced CNN-based model tailored for real-time object detection, classification, and localization.
- **Pros**:
    - **Region Proposal Network**: Efficiently suggests potential object locations.
    - **End-to-End Training**: Simplifies the training process by merging steps.
    - **High Precision**: Capable of detecting multiple objects with high accuracy.
    - **Flexible**: Can be integrated with other architectures like ResNet.
- **Cons**:
    - **Complexity**: Multiple components add to the complexity of the architecture.
    - **Memory Consumption**: Utilizes more memory due to region proposals.
    - **Slower Inference**: Not as fast as single-shot detectors like YOLO.
    - **Requires more labeled data**: For accurate region proposal and detection.
- **Research References**: [7] [8]

3. YOLO (You Only Look Once)

- **Description**: Ultra-fast object detection system that treats detection as a regression problem.
- **Pros**:
    - **Speed**: Designed for real-time applications with impressive FPS.
    - **Simplicity**: Single-shot detector, without region proposals.
    - **Unified Model**: Detects and classifies in one forward pass.
    - **Scalable**: Variants available (e.g., Tiny-YOLO) for devices with limited computational capabilities.
- **Cons**:
    - **Precision**: May have reduced accuracy compared to two-stage methods like Faster R-CNN.
    - **Spatial Constraints**: Due to grid system, struggles with small object detection.
    - **Overlapping Objects**: Might struggle when objects overlap substantially.
    - **Limited Bounding Boxes**: Predicts a fixed number of bounding boxes based on grid size.

- **Research References**: [9] [10]

Comparison Table:

Criteria	CNNs	Faster R-CNN	YOLO
Accuracy	High	Very High	High
Speed	Medium	Medium	Very High
Resource Requirement	High	Very High	Medium
Training Data Dependency	High	High	Medium
Implementation Complexity	Medium	Medium	Low

Conclusion:

Given the requirements of the UAV competition which include real-time processing, limited computational resources on-board the UAV, and the need for high accuracy, the trade-off between speed and accuracy becomes paramount.

Recommendation: Begin with **Faster R-CNN** due to its balance of speed and accuracy. If computational resources or implementation prove to be significant barriers, consider transitioning to **YOLO** for its efficiency and real-time capabilities.

2.7.1.1 Training YOLO models

To train custom dataset on YOLOv5, we need an image file (png or jpeg) and a .txt file that contains the class value, x-coordinate, and y-coordinate of the center of the target, and the width and height of the target. YOLO does not care about the bounding box drawn on the images, it just needs the annotation for each image. The coordinates, width, and height all need to be normalized to be between 0 and 1. The way to do it is dividing the x-coordinate and the width with XMAX and the y-coordinate and height with YMAX, where XMAX is the total width of the image minus one and YMAX is the total height of the image minus 1. So if we have an image of size 100 pxl x 200 pxl and the center of the bounding box at (x,y) = (26,77) and width and height of the bounding box is (w,h) = (5,6), then we will divide 26 by 99 and 5 by 99 as well, and we will divide 77 by 199 and 6 by 199 as well. We call these annotations, and it will look something like class (This is the class, it is the class number that this image has in the picture. Class number must start from 0, if not you will have to add that many empty values in the yaml

file, which we will discuss later) x-coordinate y-coordinate width height (ex: 0 0.815 0.777 0.0345 0.0392)

This phase was only for testing to make sure that we generate proper data that can be used to train YOLO, we will be creating more data and training YOLO in the next few weeks. For now, we only created all the possible color and alphanumeric combinations of circle (one of the shapes 8 shapes we need to be able to recognize). The dataset came out to be about 24,000 images, plus an additional 24,000 annotations for those images. However, 23,000 images is a lot and would take a lot of time for just testing purposes, so, trained the model on 750 training images plus 500 validation images, good time to mention how YOLO takes the data to train.

To train YOLO on custom dataset, you need to divide the dataset into three sections (by create three folders within the dataset folder), train, val, and test (train and val are mandatory, test can be ignored but good practice is to have it) each with two subfolders, images, and labels. The train folder will contain the most images and corresponding annotations, the val (validation) folder will contain less data but it will be validated after every epochs (number of times to train the model on the training dataset), and lastly test will contain even less data but will be ran for testing at the end. It is important to note that images in all these folders should be different, especially train and val, we want to validate the trained model on a unique set of images so we can increase its accuracy. We will also need a yaml file that will contain the location of each folder, train, val, and test, in addition to number of classes and the names of classes. It would look something like this:

```
train: ../dataset/train
val: ../dataset/val
test: ../dataset/test

nc: 8 // number of classes

names: ["", "", "", "", "", "", "", "Circle"]
```

In the example we can see that Circle is placed on position 7 (start from position 0) of the array because in our annotations we set the class to 7 instead of 0. Either way, we will place this file in the data folder, which is within our yolov5 cloned folder. We need to place the dataset folder in the yolov5 cloned folder as well. To start training, you need to navigate to the yolov5 cloned folder in command prompt, and then place this command: python train.py --img-size 640 --batch-size 16 --epochs 50 --data data\your_yaml_file.yaml -- weights yolov5s.pt.

The YOLO model can now detect very small circles on a picture. The training time for 1250 images was more than 5 hours, and we have a dataset of more than 24,000 images per each class, training the model on this sized dataset (which we should increase accuracy) will require some high specs, especially a good GPU. However, there is a bit of a problem with the dataset which we need to fix, even though we were able to train the model and it can detect small images, it is not detecting large circles. The next step is to analyze the results and look for imperfections, there are results that YOLO generates that we can use to analyze the results.

3. Reflection/Method Revisions

Various reflections and method revisions:

- Consider Nelson Brown, Armstrong engineer, who is currently working on dataset collected in a variety of altitudes, but may have aerial images that might be within the ideal images for data collection.
- Due to the recruitment phase not starting earlier in the summer it led to a reduced amount of time to effectively work on the project and the software team had to deal with recruitment and retention of members in the beginning of the semester.
- The lack of expertise in the team was attempted to be remedied by overviewing the relevant libraries, but still needed more fundamental understanding. Although there are classes that teach machine learning applications, it does not say for certain that it gives the ‘right’ fundamentals for the project’s scope. As the classes attend to teach the overarching domain of machine learning applications and not specifically into computer vision and object detection classification.

4. Conclusion/Future Work

During the fall semester, was able to recruit and get members to have a basic understanding of machine learning and libraries that are used to implement a machine learning model. Due to the group having only a simple understanding of ML and no experience in machine inference implementation, this semester was mainly focused on research and data collection for the models. Was able to determine what is going to be used for data collection for standardized target shape, color of alphanumeric & shape, and the alphanumeric. Then using YOLO architecture for the shape model to have fast inference time in order to do real time machine inference. Then the shape model will likely use YOLO series, but remains to be determined during the winter break. For the alphanumeric predicting model, would be using openCV OCR to detect text and output the found texts. For obstacle avoidance, it was pure research for the semester as knowledge and implementation is new and as of what is known for curriculum is only within the grad student classes.

For future work, once data collection has been completed and the models for each dataset have been created. Would need to make the models to be able to run on a Jetson Orin Nano Developer kit. Would attempt to turn the models into TensorFlow lite models, or equivalent form, to be ran on the developer kit. Benchmarks of the FLOPS of the models and the various settings of the developer kit “AI performance” would be tested and implemented. Then to fix issues of lack of expertise for future SUAS software members, would begin recruitment in SPRING 2024, to allow new members to have more time to familiarize themselves with the work and be capable of contributing in FALL 2024. Also reduces the need for recruitment for the next SUAS competition and utilizes the full semester to work rather than recruiting again. Then also consider making a whole team entirely designated for obstacle avoidance as the scope of team leader becomes too wide to cover all bases when asked to have specialty in two different topics.

5. References

- [1] https://sites.google.com/view/grli-uavdt/首页
- [2] https://vision.ee.ccu.edu.tw/aerialimage/
- [3] https://cemse.kaust.edu.sa/ivul/uav123
- [4] https://www.proquest.com/docview/2530133784/fulltextPDF/8A0658D0D93F45C0PQ/1?accountid=10357
- [5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Advances in Neural Inf. Process. Systems*, 2012, pp. 1097-1105.
- [6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Advances in Neural Inf. Process. Systems*, 2015, pp. 91-99.
- [8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2016.
- [9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2016, pp. 779-788.
- [10] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2017.