PennAiR Aerial Robotics

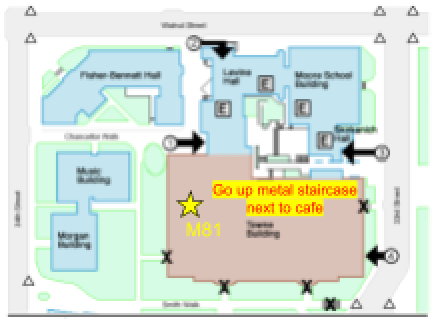
SOFTWARE CHALLENGE (FALL 2024)

Overview
The PennAiR Software Team consists of two subteams – Vision and Navigation/Controls – that work together to program the aircraft. The vision subteam utilizes machine learning that allows the aircraft to identify key ground markers set by the competition organizers. The Navigation/Controls subteam enables the aircraft to be remotely controlled as well as implement autonomous routing algorithms. Team members have primary assignments on one of these subteams but also work closely with members on other subteams.

This year's fall application timeline will be in two waves. The **priority wave** is for those with a strong passion in the aerospace industry and want to be involved with team activities as soon as possible. The priority deadline is early, but a good application by this deadline also demonstrates your strong interest in the club. We also understand that there are many clubs to choose from and you may want a little more time to explore. Because of this, most students will opt to apply during the **regular wave**, which is due a bit later. It is important to note that there is no "cap" to the number of members on the team, and our number has fluctuated from year to year. Historically, we have tried to be as inclusive as possible (while being limited by available resources). Applying during the priority wave will give you a slight advantage simply because there are less applicants in the pool, but we do expect many members to be accepted during the regular wave. (This is also designed to help our leads by spreading out the applications 😊)

	Priority Wave	**Regular Wave**
Application Due	September 7th, 11:59 pm	September 20, 11:59 pm
Interviews	September 9-12th	September 23-26th
Notification	September 13th	September 27th

Application Details (General Questions on Google Form + Challenge Below)
The challenge below is designed to be difficult, which allows us to see how you think as an engineer. We are not expecting fully correct answers, but rather want to see effort and a desire to learn more. Should you have any questions, feel free to email board@pennaerial.com or come to office hours on September 5th and 18th between 7-9 pm in Towne M81 (see map below)



Last year, as part of the SAE Aero Design Competition, we had to come up with a vision and control algorithm to autonomously detect and land within a marked region on a field. Now, you guys can try a simplified version yourselves!

Objective:

Develop an algorithm to detect solid shapes on a grassy background, trace their outlines, locate their centers, and apply the algorithm to a video. For extra credit, make the algorithm agnostic to background color and shape color gradients. You should attempt to get as far in the challenge as you can, but we are not necessarily expecting all parts to be completed.

Task Overview:

1. **Part 1: Shape Detection on Static Image:**
   - Use this image (PennAir 2024 App Static.png) that features solid shapes on a grassy background.
   - Implement an algorithm (we recommend OpenCV - cv2) to detect the shapes.
   - Trace the outlines of the detected shapes.
   - Locate and mark the centers of the shapes.
2. **Part 2: Shape Detection on Video:**
   - Apply the algorithm to the video file (PennAir 2024 App Dynamic.mp4).
   - Ensure the algorithm consistently detects, traces, and locates the centers of the shapes throughout the video.
   - Please treat the video file as a streamed input - this part should involve applying your prior algorithm to each frame. Just like how our aircraft doesn't have the entire video journey ahead of time, we want to feed the algorithm each frame one at the time.
3. **Part 3: Background Agnostic Algorithm:**
   - Modify the algorithm to work with various background colors and textures.
   - Ensure the algorithm maintains its accuracy in detecting, tracing, and locating shapes regardless of the background.
   - Test with (PennAir 2024 App Dynamic Hard.mp4)
4. **Part 4: Make it 3D (optional)**
   - You may have realized that this has been a 2D task, but the task should actually be in 3D. Extend your algorithm to instead mark the depth, x, and y coordinates of the centers w.r.t the camera. Assume that the camera's intrinsic matrix is K=[[2564.3186869,0,0],[0,2569.70273111,0],[0, 0, 1]], and that the circle has a radius of 10 in. For simplicity, assume a flat surface.
5. **Part 4: Anything else:**
   - If you have any improvements/extra enhancements in mind (tracking objects even through overlap, etc.), implement them! We're excited to see what you come up with.

For reference, here is an example of something that accomplishes everything up to Part 3:
pennair_2024_sample sol.mov

Deliverables:

1. **Code Implementation:**
   - Provide the source code for the implemented algorithm.
   - The code should be well-(enough)-documented and include comments explaining the key steps and logic.
2. **Static Image Results:**
   - Submit the processed image showing detected shapes with their outlines traced and centers marked.
   - Provide a brief explanation of your approach and any challenges faced.
3. **Video Results:**
   - Submit a screen recording of the processed video, demonstrating the algorithm's performance.
   - Include a brief report on how the algorithm performs and any adjustments made to improve it (e.g. using less computationally-expensive operations).
4. **Background Agnostic Results:**
   - Again, the processed video demonstrating the algorithm's performance.
   - Include a brief discussion on the modifications made to achieve background agnosticism.

Submission Instructions:

1. **Repository:**
   - Create a GitHub repository for your project.
   - Include all source code, processed images, videos, and reports in the repository.
   - Ensure the repository is public and share the link in the google form field.
2. **Documentation:**
   - Include a README.md file in the repository with clear instructions on how to run your code.
   - Provide any additional documentation required to understand and evaluate your solution.
   - Addendum: it would be greatly appreciated if you could also embed the videos in your README file(s) 🙂

Evaluation Criteria:

1. **Accuracy:** Precision in detecting, tracing, and locating shapes.
2. **Efficiency:** Performance of the algorithm in real-time processing for videos.
3. **Robustness:** Ability to handle different backgrounds and maintain accuracy.

4. **Code Quality:** Clarity, documentation, and organization of the code.
5. **Innovation:** Creative approaches to solving the challenge and handling potential issues.

We look forward to seeing your innovative solutions and considering you for Penn Air's software team!