# AI-Controlled Deployable Vortex Generators for Adaptive Airflow Optimization on Airfoils

**Ishan Kasam**
**Massachusetts Academy of Math and Science**
**STEM Project**
**Instructor: Kevin Crowthers, Ph.D.**

**Table of Contents:**

# Good Laboratory Practice
# Record-Keeping Contract

**I commit to record keeping in accordance with Good Laboratory Practices.**

- My experiments and records will be reproducible, traceable, and reliable.
- I will NOT write my notes on scraps of paper, post-it notes, or other disposable items. My notes will go directly into my laboratory notebook.
- My data will be recorded in real-time. If I cannot record data in real-time, I will record raw data as soon as physically possible.
- I will record both qualitative and quantitative observations in my laboratory notebook and laboratory reports.
- My laboratory notebook will include information on the materials and instruments utilized during experimentation.
- I will initial and date (with time) over the edge of any material that is taped or pasted into my laboratory notebook.
- I will provide a real-time record of any analysis I perform.
- I will use blue or black pen to make entries in my laboratory notebook. I will NOT use a pencil.
- I will define ALL abbreviations.
- If I make a mistake in my laboratory notebook, laboratory worksheets, or other written material, I will not obliterate or obscure the mistake. Instead, I will cross out the mistake using a single line. Any empty spaces in tables or partially used notebook pages will be crossed out using a single diagonal line.
- If I record information online (ex. In Google Drive), I will log in so that my contributions are traceable.
- I will initial and date each page in my notebook and the front of each laboratory report.

**Ishan Kasam**
Printed name

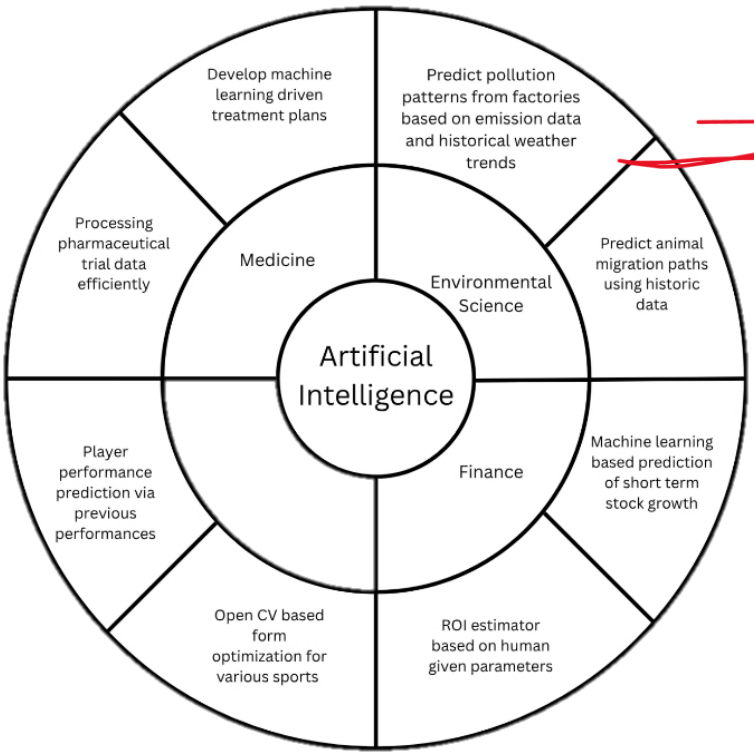
**Ishan Kasam**

Signature

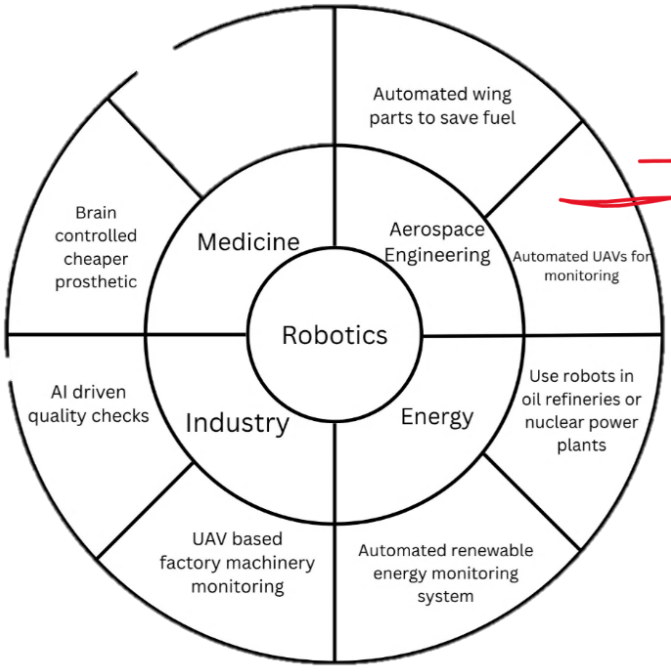
Date 9/5/2025

# Brainstorming- provide a figure caption, sign and date any figures

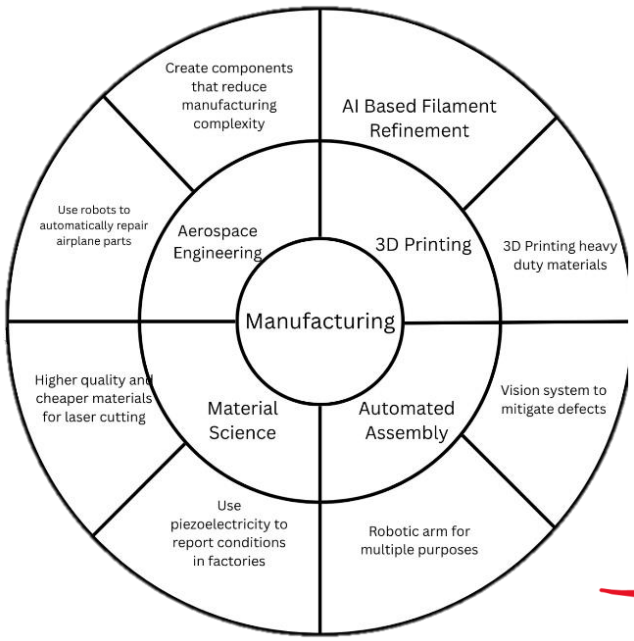
## Pie Diagrams:




Idea pie chart for ideas focused on artificial intelligence 8/23/2025



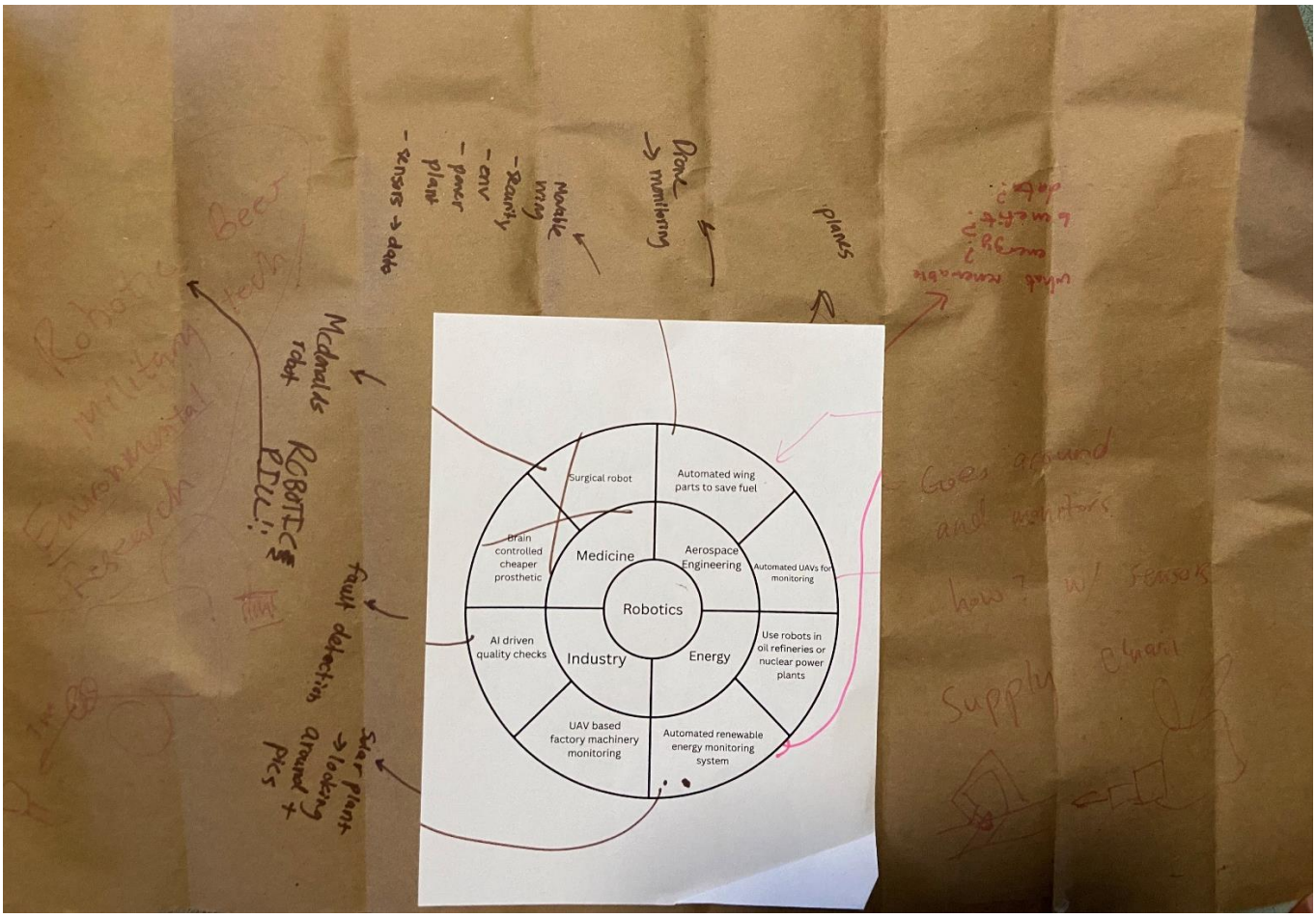
Idea pie chart for ideas focused on robotics 8/23/2025

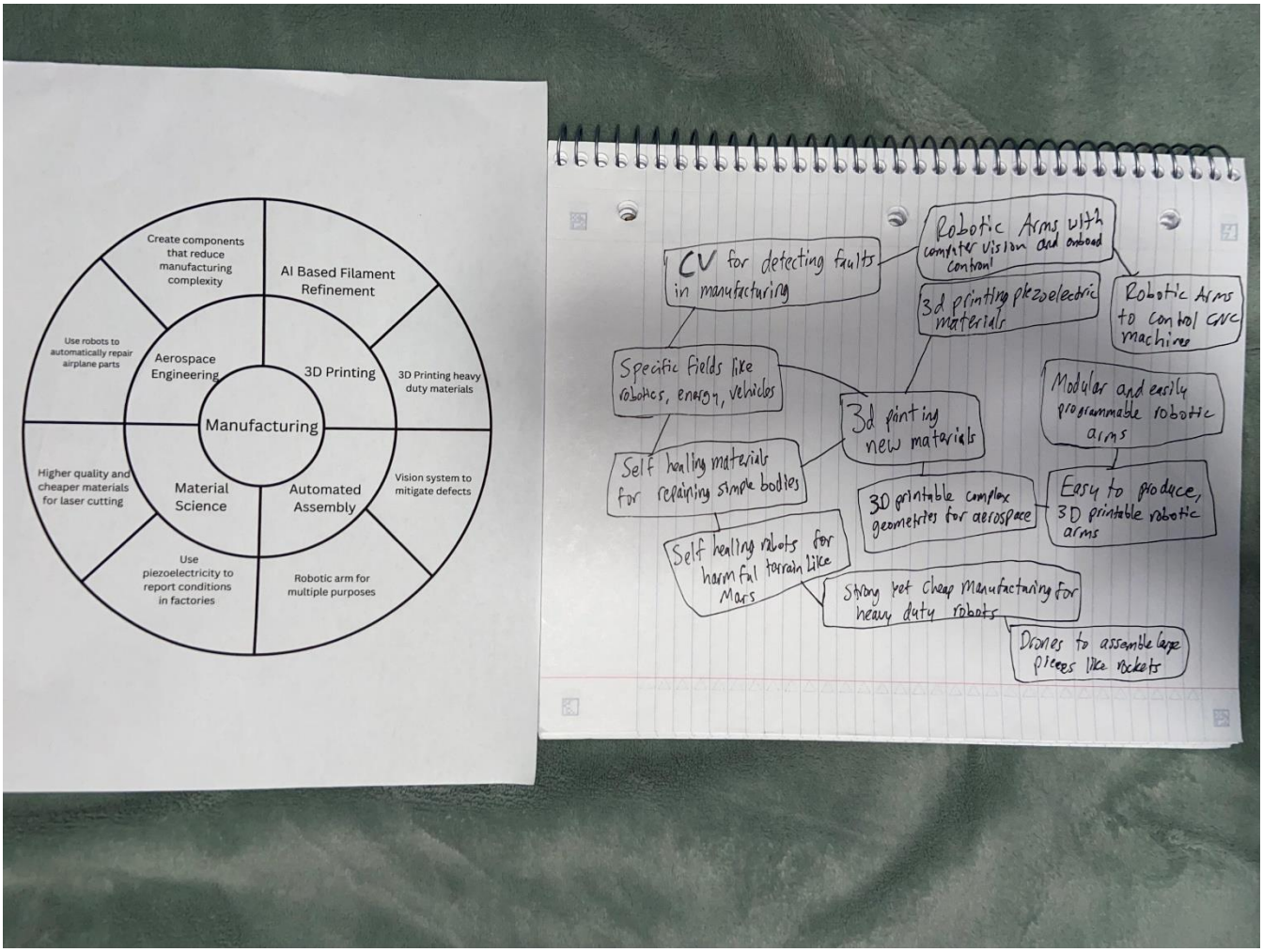



Mind map done in a group brainstorming session. The ideas are focused on robotics. 9/29/2025

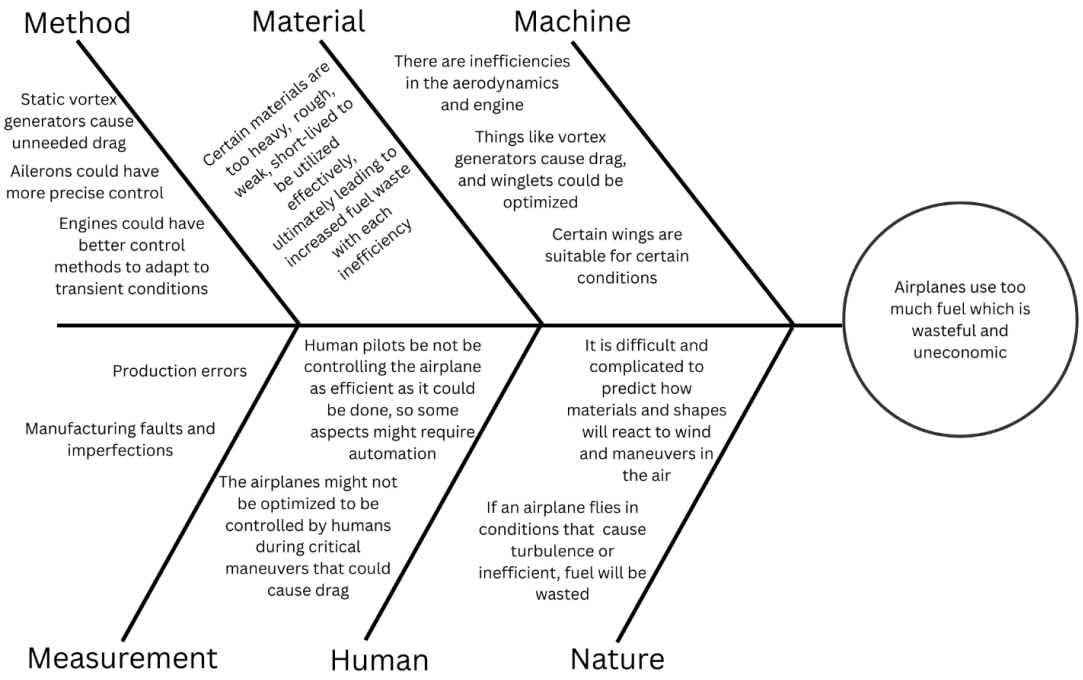
Idea pie chart for ideas based on manufacturing innovations 8/23/2025Mindmaps:

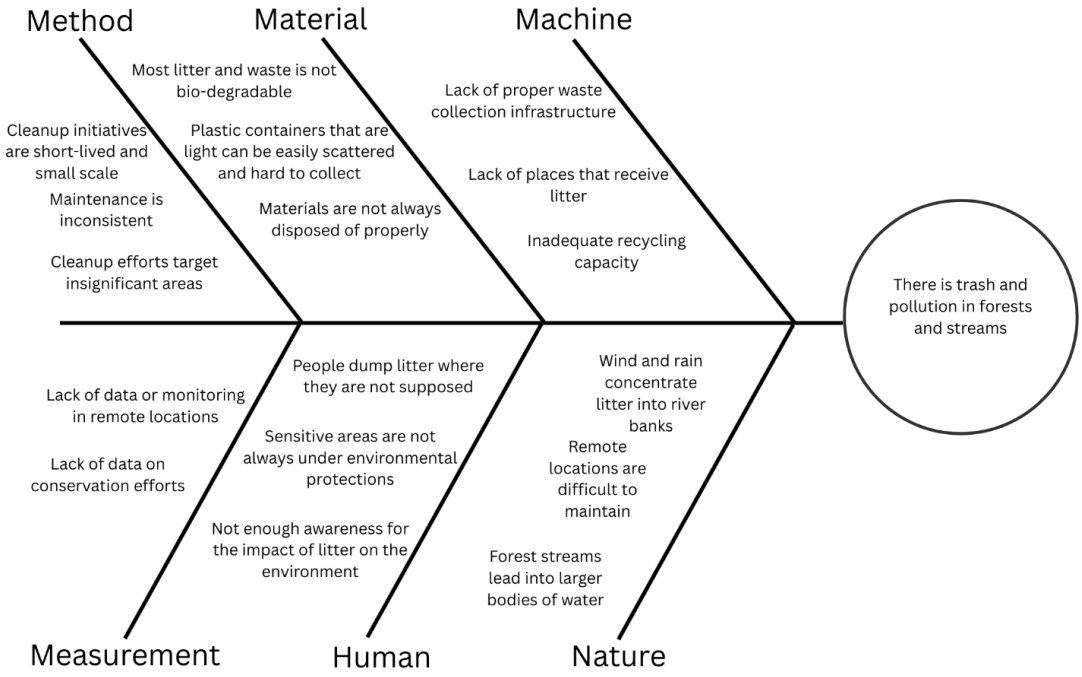
Mind map done in a group brainstorming session. The ideas are focused on AI and ML. 9/29/2025


Robotics
Medicine
Aerospace Engineering
Industry
Energy
Surgical robot
Automated wing parts to save fuel
Brain controlled cheaper prosthetic
Automated UAVs for monitoring
AI driven quality checks
Use robots in oil refineries or nuclear power plants
UAV based factory machinery monitoring
Automated renewable energy monitoring system
Drone monitoring
planes
Mcdonalds robot
fault detection
Goes around and watches
how? w/ sensors
Supply
Share

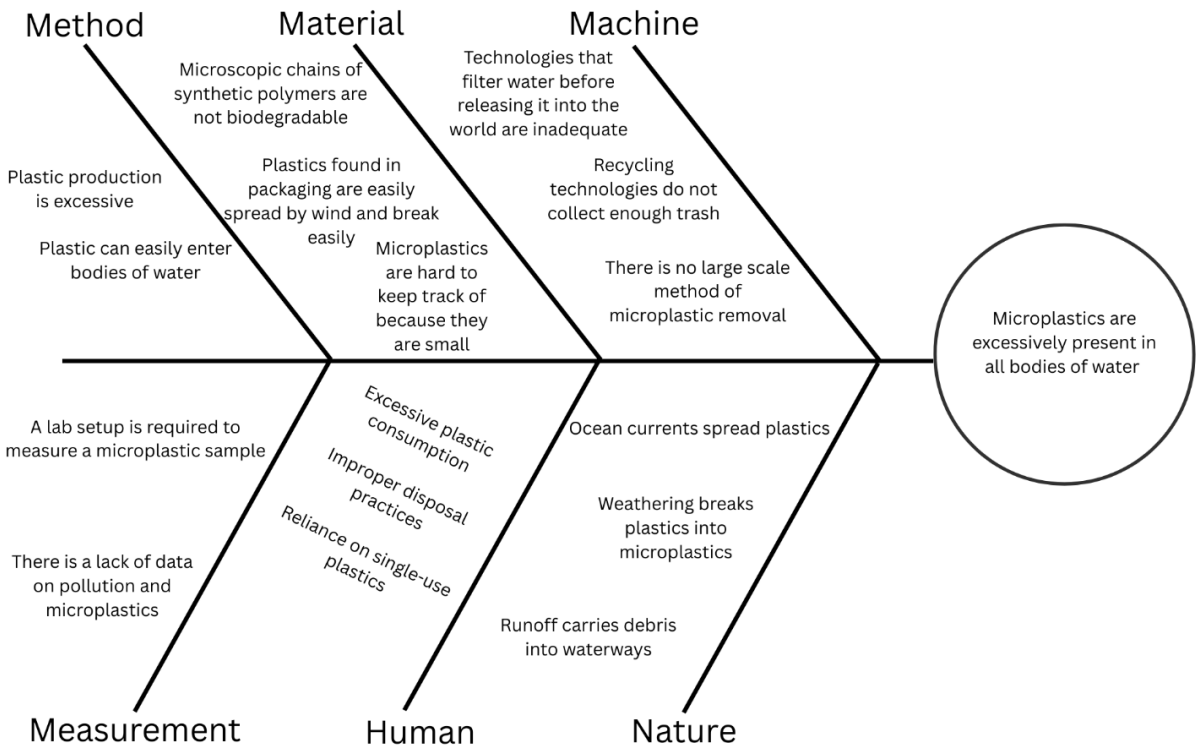



Mind map done in a group brainstorming session. The ideas are focused on manufacturing. 9/29/2025




Fishbone Diagrams:




## 5 Whys:

Airplanes use too much fuel

1. Why do airplanes use too much fuel?
    a. Jet engines need to burn through a lot of fuel to get an adequate amount of thrust.
2. Why do they need to go through a lot of fuel to get enough thrust?
    a. Current engine designs and aerodynamic designs focus on reliability over efficiency.
3. Why is there is a large focus on reliability?
    a. Developing novel systems for aerodynamics and engines can be expensive and risky.
4. Why is creating new technologies expensive and risky?
    a. There are high amounts of certification and testing to be done before moving parts can be implemented onto an airplane.
5. Why is this certification needed?
    a. Aircrafts are delicate and sensitive, so small failures could lead to large consequences.

Trash and pollution in forests and forest streams

1. Why is there trash in forests and forest streams?
    a. Waste makes its way into natural areas and does not get picked up.

2. Why does waste in natural areas not get picked up?
    a. Those areas are hard to reach and small scale efforts will not impact large areas.
3. Why do those areas need more time and effort to maintain?
    a. Those areas are difficult to access, so more people and resources will be needed.
4. Why are more resources needed?
    a. Collecting waste manually is slow, requires many people, and will not be a consistent effort.
5. Why is collecting waste manually slow and inconsistent?
    a. Remote areas are hard to routinely access and maintain.

Microplastics are abundant in bodies of water

1. Why are microplastics abundant in bodies of water?
    a. Plastic products break down into tiny fragments in natural environments
2. Why do plastic products break down into tiny fragments in natural environments?
    a. Plastics are often in rivers or lakes which can break plastic into microscopic parts.
3. Why is there large amounts of plastic in rivers and lakes?
    a. Plastics are widely used and often discarded without being properly recycled or contained.
4. Why are plastics discarded without being properly recycled or contained?
    a. Plastics are easy to replace and have no value, so it is easy to mistreat them, and this causes them to accumulate in rivers and streams.
5. Why are there not more conservation efforts in cleaning microplastics out of bodies of water?
    a. Microplastics are very hard to track and get rid of once they enter the environment.

Develop deployable vortex generators with a machine-learning control system that adjusts height in real time based on sensor input.

Brief Overview

This project proposes an alternative to static vortex generators by developing deployable vortex generators with a machine learning-based control system. While static vortex generators effectively delay stall and enhance post-stall behavior by energizing the boundary layer, they introduce parasitic drag during cruise conditions when separation control is unnecessary, reducing overall aircraft efficiency. This project addresses this limitation through an adaptive deployable vortex generator system using machine learning control trained on computational fluid dynamics (CFD) simulations to optimize deployment height in real-time based on instantaneous flow conditions.

Engineering Design Brief or Research Outline

# Professional Communication:

On 8/26/2025, I sent the following email to Professor Calli Berk of Worcester Polytechnic Institute: Dear Professor Calli,

My name is Ishan Kasam, and I am a student at the Massachusetts Academy of Math and Science at WPI with a strong interest in robotics. As part of my education at Mass Academy, I am pursuing a long-term (5–6 months) science fair research project, and during my search for literature and labs, I came across your work in the MER Lab.

I am currently exploring a project idea for a caterpillar-inspired robot that uses this type of locomotion. The robot would employ machine learning–based vision to detect trash, LiDAR to sense depth, and mapping algorithms to navigate unstructured forest environments. It would be equipped with a manipulator capable of gripping and suction-based picking, allowing it to collect trash efficiently. The goal is to develop a system capable of picking up trash in natural settings and supporting forest conservation.

In reviewing your work, I was particularly drawn to your research on robotic waste sorting, focusing on manipulation algorithms for picking and rearranging objects, as well as datasets for waste recognition. Your efforts in enhancing robotic manipulation in unstructured environments closely align with the challenges my project aims to address. The use of machine learning–based vision for object detection, combined with a gripping and suction-based manipulator for trash collection, mirrors your lab's objectives in automating waste handling.

If you are available, I would greatly appreciate the opportunity to meet with you online for a short call (10-15 minutes) to discuss my project idea in more detail. I am specifically seeking your feedback on the concept, potential challenges, and any guidance you could offer to help refine the approach.

Thank you very much for your time and consideration.

Sincerely,
Ishan Kasam

On 9/11/2025, I spoke on a call for 30 minutes with Professor Calli for advice on how to approach my project idea as well as how to refine the idea. I also discussed the projects that are ongoing in his lab, and I received suggestions on how to manage the network and physical design of the robot.

On 9/2/2025 I sent the following email to Professor Jayachandran of Worcester Polytechnic Institute

Dear Professor Jayachandran,

My name is Ishan Kasam, and I am a student at the Massachusetts Academy of Math and Science at WPI. I am at the initial stage of exploring a long-term science fair research project on deployable vortex generators, where MEMS actuators adjust VG height in real time using sensor and CFD data, guided by an AI model predicting boundary layer state and determining optimal deployment. I am seeking guidance on using CFD to help model rapid VG deployment under transient aerodynamic loads.

I recently read your article *Effect of Unsteady Pressure Rise on Flame Propagation and Near-Cold-Wall Ignition*, and I noticed your approach to modeling transient pressure effects. This struck me because, in my project, the vortex generators will induce rapid, localized changes in flow.

I plan to simulate deployment in CFD to predict transient boundary layer response and use that data to guide the AI-based control of the actuators. I would greatly appreciate your thoughts on this approach. Does it seem reasonable, or are there key considerations I might be overlooking?

I would be grateful for any guidance you could provide, whether by email or a brief Zoom meeting.

Thank you for your time,
Ishan Kasam

On 9/16/2025, I spoke on a call with Professor Jayachandran about how to approach the CFD testing phase for attaining the training data. I learned that I could possibly use some of the WPI resources to run simulations as well as wind tunnels.

Wed, Oct 8, 2025, I sent the following the email:

Dear Professor Lozano-Durán,

My name is Ishan Kasam, and I am a high school junior at the Massachusetts Academy for Math and Science. I am in the initial planning stages of a research project on deployable vortex generators controlled by AI for aerodynamic flow control for low to high Re. I am reaching out because of your expertise in turbulence modeling and high-fidelity simulations.

I have copied my teacher onto this email.

My project focuses on modeling boundary-layer separation, evaluating the performance of deployable vortex generators at different heights under varying flow conditions, and using the simulation data to train a machine learning model. I am looking for guidance on turbulence modeling strategies and CFD workflow practices that can capture unsteady separation while remaining computationally efficient.

Would you be open to further discussion for offering advice or suggesting resources that could help me refine my simulation approach?

Thank you very much for your time.

Sincerely,
Ishan Kasam

# Materials and Methods:

CFD simulations will be run on a NACA 4415 airfoil across a range of Reynolds numbers, angles of attack, and VG heights. Data on pressure, lift, and drag will be recorded. Adequate mesh resolution will be used with a y+ that is less than 1. For preliminary testing, RANS k-omega will be used, but for the machine learning training data, URANS k-omega will be used to increase precision.

A small neural network will be trained with Tensorflow/Keras. The model will take inputs of flow speed and static pressure, and it will output optimal VG height. It will be trained on the full CFD dataset, where each case includes the pressure inputs and flow speeds, and the known VG height that caused those readings.

A physical model of the airfoil equipped with pressure sensors across the span, an actuator to move the vortex generators, rectangular vortex generators linked by a rigid carbon fiber rod, a Raspberry Pi 4 microcontroller, and a battery for a power source.

After loading the trained neural model onto the Raspberry Pi, the complete airfoil system will be tested in the wind tunnel. These tests will then be used to run a reinforcement-learning stage, allowing the controller to adjust its policy and reduce the gap between the pretrained (simulation-based) model and the actual tunnel behavior.

# Daily Entries:

## Experiment 1: Effects of Vortex Generators of Different Heights at Re = 1,500,000, 12/7/2025, Ishan Kasam

**Introduction**: This is a test for preliminary data to show that airfoils with VGs of different heights have different impacts on Cl, Cd, and Cp at different angles of attack, and it is to show that at different points in a flight envelope, different height VGs are beneficial.

**Methodology**:

- I downloaded the baseline airfoil coordinates from Airfoil Tools and imported them into SolidWorks.
  - The airfoil was extruded to a span of 111.8 mm.
  - I modeled counter-rotating triangular vortex generators and placed them on the suction surface at 10% chord.
  - Two VG heights were created: a medium height of 1.34 mm and a larger height of 2.81 mm.
- The fluid domain was created in ANSYS DesignModeler.
  - I used Bodies of Influence around the airfoil and near wake so the mesh could be refined only where needed.
  - This kept the cell count lower in the far field while still capturing important flow features.
- Meshing was done in ANSYS Meshing using an unstructured mesh.
  - Inflation layers were added around the airfoil and VG surfaces.
  - The goal was a y+ value below 1 to properly resolve the viscous sublayer and boundary layer behavior.
- The CFD simulations were run in ANSYS Fluent using a pressure-based steady RANS solver.
  - The SST k-omega turbulence model was used because it performs well for separation under adverse pressure gradients.
  - Curvature correction was turned on to better capture vortex-induced streamline curvature.
- Boundary conditions were set up as follows.
  - Velocity inlet based on Re = $1.5\times10^6$ at angles of attack of 4° and 16°.
  - Turbulence intensity set to 0.1% to represent clean wind tunnel flow.
  - Pressure outlet set to 0 Pa gauge pressure.
  - Side boundaries set as translational periodic to simulate an infinite span.
  - Top and bottom boundaries set as symmetry conditions.

- Solver settings were chosen to improve stability.
    - The coupled scheme was used for pressure-velocity coupling.
    - Pseudo-transient time stepping was enabled to prevent early divergence while vortices formed.
    - Gradients used least-squares cell based.
    - Pressure, momentum, and turbulence equations used second-order upwind discretization.
- Convergence was judged based on the aerodynamic coefficients, not just residuals.
    - The simulation was considered converged once lift and drag stayed flat for at least 200 iterations.
    - A small mass flow imbalance between inlet and outlet was also required.

**Observations and Experimental Data**: prelim_results.xlsx

This data seems to match my predictions that different vortex generator heights would yield different results in terms of efficiency, but the results for each case in the NREL Wind tunnel study have at least a 20% error, but the general trend makes sense.

**Calculations and Data Analysis**: Data was analyzed and turned into bar charts in Excel as can be seen in the data file.

**Concluding Remarks**

Something to change for the future would be increasing the domain size, reducing the mesh size, using better solver settings, and properly setting a periodic boundary condition, as there was a software glitch with setting it up the first time. This will serve as good preliminary data to show a trend to be further explored.

# Experiment 2: Choosing Test Cases for Low Fidelity CFD Dataset, 12/24/2025, Ishan Kasam

Introduction: This is a test for generating test cases for my low fidelity test cases to then use in my Co-Kriging machine learning algorithm. These test cases will be used to automate CFD simulations. My primary focus today was defining the simulation parameters that will serve as the "ground truth" for the project. Since running high-fidelity CFD is computationally expensive, I can't just simulate every possible combination. I needed to construct a dataset that is lean but statistically significant enough to train the multi-fidelity Co-Kriging model later on.

The specific goal was to create two main arrays: the physical geometry changes (VG heights) and the flight conditions (Angle of Attack and Reynolds Number). I needed the VG heights to capture different interactions with the boundary layer, specifically, cases where the actuator is submerged within the boundary layer, cases where it peeks out, and cases where it is fully exposed to the free stream to energize the flow during deep stall.

Methodology:

- I picked a set of discrete VG heights based on estimated boundary layer thickness.
  - The range was 0.75 mm to 3.00 mm.
  - 0.75 mm represents a very small deployment mainly for cruise efficiency.
  - 3.00 mm represents the maximum deployment meant for stall recovery.
  - The middle values (1.25, 1.75, 2.25 mm) were added so the model can learn the nonlinear transition between small and large deployments.
- For the flight conditions, I intentionally did not use a simple linear sweep (like every 2° of angle of attack).
  - That kind of sweep wastes simulations in simple regions and misses the complicated stall behavior.
  - Instead, I used Latin Hypercube Sampling (LHS).
- I wrote a script that treats the flight envelope as a 2D design space.
  - Angle of attack range: 0° to 20° to cover attached flow through deep stall.
  - Reynolds number range: about 0.8 to 2.5 million to capture realistic speed changes.
- I used the LHS algorithm to generate 22 operating points inside this space.
  - The points are semi-random but evenly spread out.
  - No two points share the same row or column, which helps maximize how much new information each CFD run provides.
- I slightly biased the sampling toward the stall region (12°–20°).
  - That is where the physics become highly nonlinear.
  - It is also where vortex generators matter the most.
- This produced a set of 22 operating points.
  - Running those across 7 VG heights gives 154 CFD cases.

- This forces the AI model to interpolate across both Reynolds number and angle of attack instead of memorizing simple curves.

Observations and Experimental Data:
https://docs.google.com/document/d/1jkC5kYG5nPduLiXg_jG9ZSu0Moxsh9_FUOebLIUbD0k/edit?usp=sharing

The test cases seem to cover a reasonable range of values, just observing optically.

Calculations and Data Analysis: The calculations were performed by the Python code. It used Latin Hypercube sampling to select the values in the array.

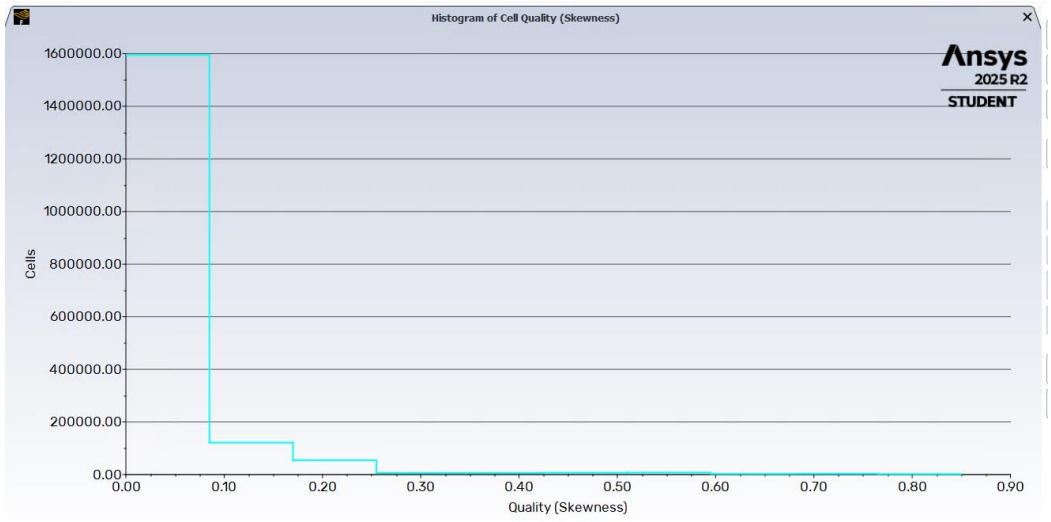
Concluding Remarks
I will use these test cases.

# Experiment 3: Low Fidelity CFD Simulations Mesh and Geometry Creation, 12/24/2025, Ishan Kasam

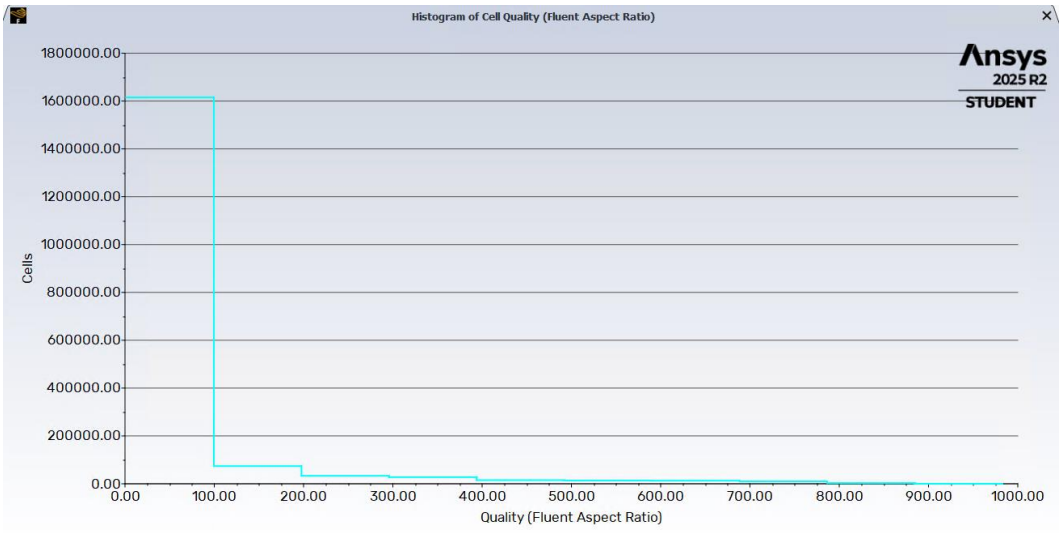
Introduction: Meshes of low cell count yet decently high fidelity near critical regions need to be made to compute quickly while still providing valuable insight as to the trends in vortex generators. There needs to be a y plus of less than 1 near the wall to resolve the vortices and turbulent structures, and I will compare a purely tetrahedral mesh vs. a poly-hexcore mesh

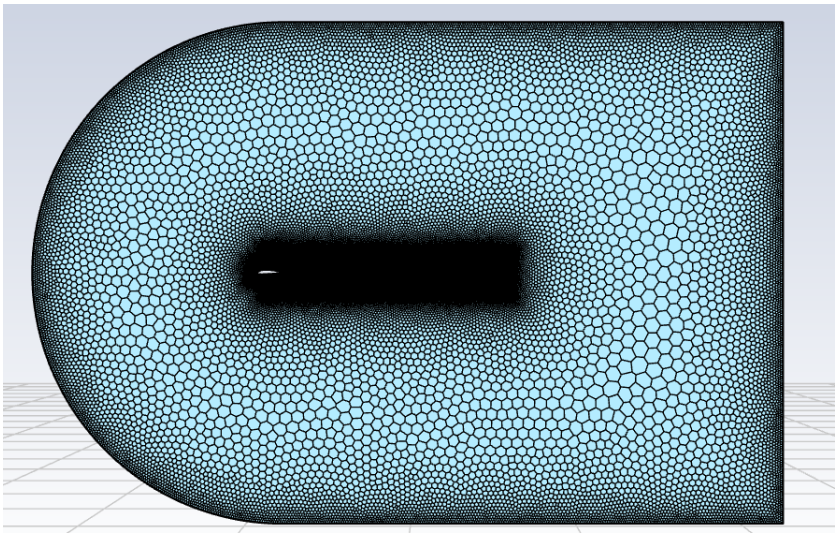
Methodology:

- 1I built seven different airfoil geometries in SolidWorks to represent the full range of deployable VG states.
    - This ranged from a clean airfoil (0 mm) up to the maximum deployment of 3 mm.
    - Intermediate heights like 0.75 mm, 1.25 mm, and 2.7 mm were included to capture how sensitive the aerodynamics are to VG height.
    - A computational domain was created around each geometry with enough distance to avoid boundary interference.
- I then ran a mesh topology study to figure out the best meshing strategy.
    - I used the 2.7 mm VG case as the test geometry since it has the most complex flow features.
    - Two different mesh types were tested.
- First option: an unstructured tetrahedral mesh made in ANSYS Mechanical.
    - These adapt well to complex geometry.
    - But they usually need more cells to reach the same accuracy.
- Second option: a Mosaic Poly-Hexcore mesh made in ANSYS Fluent Meshing.
    - This uses mostly hexahedral cells in the bulk flow.
    - A layer of polyhedral cells connects the hex core to the inflation layers near the walls.
    - The goal was better mesh quality and faster solving with fewer cells.
- I ran preliminary CFD simulations using both meshes with identical boundary conditions.
    - Lift and drag were compared against experimental wind tunnel data from the NREL S814 study.
- The Poly-Hexcore mesh matched the experimental data better than the tetrahedral mesh.
    - It also achieved this accuracy with fewer cells and better orthogonal quality.
    - That meant lower computational cost without losing accuracy.
- Based on this, I selected the Poly-Hexcore mesh for the entire project.
    - The meshing workflow was then automated and applied to all remaining geometries.
    - This kept mesh quality, boundary layer resolution (y+ > 1), and grid density consistent across the dataset.




.



Examples from 3mm VGs
Orthogonal

Minimum = 0.15008259, Maximum = 1, Average = 0.95480316.

Skewness
Minimum = 0, Maximum = 0.84991741, Average = 0.033449709

Aspect Ratio
Minimum = 0, Maximum = 0.84991741, Average = 0.033449709

Average cell count is 1.765 million for each geometry with vortex generators, and it was 1.45 million cells for the plain airfoil mesh.

Concluding Remarks: Optically, the mesh seemed to have a good distribution of small cells near the airfoil, and the transition from small cells in the near-wake region to the slightly further wake region to the far field looked like it would perform well, but there might be an issue with backflow on the pressure outlet. The cell count is at a very good number, and each simulation should run in around 1 hour.

# Experiment 4: CFD Validation 12/2/2025, Ishan Kasam

**Introduction**

This experiment was performed to verify that the CFD setup produces stable, mesh-independent, and physically reliable aerodynamic results before generating the full dataset. The goal was to confirm that the mesh density, domain size, and wake length were large enough to prevent numerical instability and artificial boundary effects. A final validation step was then performed by comparing simulation results against published experimental wind-tunnel data for the NACA 4415 airfoil with vortex generators. This experiment establishes confidence that the low-fidelity CFD dataset is suitable for machine-learning training.

- **Methodology**
  I ran a mesh independence study using the 2.71 mm vortex generator case at Re = 1.5 × $10^6$ and an angle of attack of 12°.
  - I generated three meshes using the same meshing strategy and inflation setup: 0.8 million cells, 1.8 million cells, and 2.8 million cells.
  - The 0.8 million cell mesh kept diverging and crashing before convergence, with strong residual growth and unstable lift and drag histories.
  - The 1.8 million and 2.8 million cell meshes both converged successfully.
  - The 1.8 million mesh gave roughly Cl = ~1.19 and Cd = ~0.130.
  - The 2.8 million mesh gave roughly Cl = ~1.19 and Cd = ~0.1298.
  - The difference between them was under 0.002% for both coefficients, which showed the solution was mesh independent.
  - Because of this, I chose the 1.8 million cell mesh for the full low-fidelity dataset to save computation time without losing accuracy.
- I then tested domain size to make sure the boundaries were not affecting the results.
  - I tried placing the top and bottom boundaries at 2c, 3c, and 5c from the airfoil.
  - The 2c and 3c domains both had backflow at the pressure outlet and would not converge.
  - The 5c domain stayed stable and gave consistent aerodynamic coefficients.
- I also tested wake length behind the airfoil.
  - I tried wake lengths of 5c, 7c, and 9c.
  - The 5c and 7c cases showed outlet instability and oscillating residuals.
  - The 9c wake length stayed stable and converged properly.
  - Final domain size was set to 5 chord lengths above and below the airfoil and 9 chord lengths downstream.
- After locking in the mesh and domain, I validated the CFD results using the Reuss et al. (1995) wind tunnel study.
  - I ran simulations at 12°, 18°, and 19° angle of attack with a 2.71 mm vortex generator at Re = 1.5 × $10^6$.
  - At 12°, CFD predicted Cl = ~1.19 and Cd = ~0.130 vs experimental Cl = ~1.15 and Cd = ~0.135 (about 3–4% error).

- At 18°, CFD predicted Cl = ~1.60 and Cd = ~0.168 vs experimental Cl = ~1.54 and Cd = ~0.175 (about 3–4% error).
- At 19°, CFD predicted Cl = ~1.68 and Cd = ~0.206 vs experimental Cl = ~1.62 and Cd = ~0.214 (about 3–4% error).
- All errors stayed below 4.5%, which confirmed the CFD setup was producing realistic results.

**Experimental Data:**
https://docs.google.com/spreadsheets/d/1IUerkcu5M_Ah39SBbjex9TEtXuV601fQ7uTgev2jIZc/edit?usp=sharing

**Concluding Remarks**
The CFD setup was verified to be stable and mesh-independent. The 1.8 million cell mesh was chosen because it matched the 2.8 million cell results while requiring less computation time. The domain size was finalized at 5c above and below the airfoil with a 9c wake length to prevent outlet instability. Validation against published wind-tunnel data showed agreement within 4.5% for lift and drag, confirming that the simulation setup produces physically realistic results suitable for dataset generation.

# Experiment 5: Practice Low Fidelity CFD Simulations, 12/28/2025,

Ishan Kasam

**Introduction**: The purpose of this experiment was to perform a single fully manual CFD run that captured every required solver action in a Fluent Journal file. This run served as the template simulation used later to build the full automation workflow described in Experiment 6. The emphasis was not on producing useable data, but on producing a perfectly repeatable sequence of solver operations that could be converted into script logic. This experiment used the 2.7 mm vortex generator configuration.

**Methodology:**

- The goal of this run was to do one fully manual CFD simulation that captured every solver action in a Fluent journal file.
  - The idea was to create a clean, repeatable template that could later be turned into automation.
  - This run was about workflow capture, not generating research data.
  - I used the 2.7 mm VG configuration because it has enough geometric complexity to represent real cases.
- I imported the mesh from the earlier Poly-Hexcore meshing work into ANSYS Fluent.
  - The solver was set to pressure-based, steady, 3-D, double precision, with absolute velocity formulation.
  - Double precision was chosen to reduce round-off error and improve stability for later automation.
  - I checked the mesh for negative volumes and confirmed boundary zones were correctly named for scripting.
- I manually defined air properties so the journal file would record them explicitly.
  - Density = 1.225 $kg/m^3$
  - Dynamic viscosity = $1.7894 \times 10^{-5}$ $kg/(m \cdot s)$
  - Entering them manually ensures reproducibility later.
- I selected the k-ω SST turbulence model.
  - This model handles boundary layer separation well.
  - Curvature correction was turned on to better capture vortex-induced streamline curvature.
  - Default model constants were kept the same for consistency.
- Boundary conditions were entered manually so every step would be captured.
  - Test case: $Re = 1.5 \times 10^6$ and AoA = 8°.
  - Freestream velocity was calculated from the Reynolds number using the chord and air properties.
  - Velocity components were decomposed into x and y using trig and entered directly.
  - Turbulence intensity = 0.1% and viscosity ratio = 10 to mimic clean wind-tunnel flow.

- Outlet and far-field boundaries were then defined.
    - Pressure outlet = 0 Pa gauge pressure.
    - Backflow turbulence values were specified to prevent early instability.
    - Top and bottom boundaries were symmetry.
    - Spanwise boundaries were translational periodic to simulate an infinite wing section.
- I set reference values using the inlet as the source.
    - Reference velocity, area, and length were entered manually.
    - This is important because automation will later change inlet velocity, so reference values must update correctly.
- I added virtual instrumentation to mimic the future wind-tunnel model.
    - Lift and drag monitors were defined using vector-based force directions aligned with the freestream.
    - This required entering sine and cosine components based on angle of attack.
    - The monitors were set to plot and export convergence histories.
- I created discrete chordwise pressure probes.
    - Static pressure reports were defined at each point.
    - Vertex-average values were written to output files.
    - These will later be used as ML inputs and for sensor placement.
- Solver settings were tuned for stability.
    - Pressure-velocity coupling used the coupled scheme.
    - Second-order upwind discretization was used for pressure, momentum, and turbulence equations.
    - Least-squares cell-based gradients were used.
    - Pseudo-transient formulation was enabled with a global time step of 0.01 s.
- I used hybrid initialization and ran the solver manually to study convergence behavior.
    - Early iterations showed oscillations from vortex formation.
    - The solution stabilized around ~150 iterations.
    - Full convergence occurred around ~250 iterations.
    - I chose 500 iterations per case to provide a safety margin for harder cases.
- Before ending the run, I manually triggered all export steps so they would be recorded.
    - Lift, drag, and pressure histories were written to report files.
    - Full case and data files were saved for future ML training.
    - I reviewed the journal and transcript to confirm all required solver actions were captured and ready for automation..

**Experimental Data and Observations:**
The scheme file generated perfectly with a lot of nonsense that recorded GUI clicks but also had high quality content such as TUI commands that can be used later.
Final scheme file with all the test cases for 2.25mmVGs:
https://docs.google.com/document/d/1zDSJe5RfKJTQmajntvlMySwL5qqfsf5N-YHUsb5q0xE/edit?usp=sharing
**Concluding Remarks:** This script is easily reproduced and can be adapted for any VG height as long as the mesh being loaded and cases are changed.

# Experiment 6: Low Fidelity CFD Simulations, 12/29/2025, Ishan Kasam

**Introduction**: This phase documents the execution of approximately 150 low-fidelity CFD simulations designed to capture the aerodynamic performance of the deployable vortex generator system across seven geometric configurations (0 mm to 3 mm) and a full range of angles of attack (0°–20°). The primary goal is to generate a robust dataset for training a Multi-Fidelity Gaussian Process (Co-Kriging) model and the eventual AI controller, while also characterizing the stall-delaying effects of the VGs. Using automated Ansys Fluent scripts, the study will batch-process these cases using a steady-state RANS solver with the k-omega SST turbulence model, with strict monitoring of residuals and force coefficients to ensure data validity for the multi-fidelity architecture.

**Methodology**:

- I ran one manual baseline simulation using the 2.7 mm VG configuration to create a template for the automation workflow.
    - The solver was set to steady, pressure-based, with the coupled scheme for pressure–velocity coupling.
    - The SST k-omega turbulence model was used because it performs well for boundary layer separation.
    - Air properties were set to standard conditions, and the inlet was defined as a velocity inlet.
    - Velocity components in the x and y directions were calculated from the target Reynolds number and angle of attack.
- I added virtual instrumentation to the model.
    - Discrete point surfaces were placed along the chord to act like pressure taps on both suction and pressure sides.
    - Lift and drag monitors were created using force vectors aligned with the freestream using sine and cosine components.
    - Vertex-average reports were assigned to the Cp tap locations.
    - Reference values were set using the inlet as the source, including reference velocity, area, and length.
    - All commands used in this setup were recorded into a Fluent journal (.jou) file.
- I converted the recorded journal commands into a Scheme automation script.
    - This allowed logic, loops, and variable changes to be added.
    - The script runs a single geometry through a test matrix of angle of attack and Reynolds number pairs using a for-each loop.
    - For each case, the script converts angle of attack from degrees to radians and calculates the velocity needed for the target Reynolds number.
    - It rotates the inlet velocity vector and recalculates lift and drag force directions to stay aligned with the freestream.

    - The reference velocity is updated for every run so Cl and Cd remain correctly nondimensionalized.
- I built a file management system into the script.
    - Each output file uses the naming format [Geometry_Name]*[AoA]*[Re].out to keep results organized.
    - The solver initializes using hybrid initialization and runs a fixed number of iterations based on the baseline convergence study.
    - After each run, report files containing convergence histories are saved.
    - Full case and data files are also saved so the complete flow field is preserved for ML training.
- After all simulations finished, I used a Python script to parse the .out files and compile a CSV dataset linking each test case to its aerodynamic results.

**Experimental Data and Observations:**
Scheme File: https://docs.google.com/document/d/1OaNDa42Mbgfl5ahfPVilCl-jCC-VtOrjhhumlRnHt-Y/edit?usp=sharing
Full data file of results from all simulations:
https://docs.google.com/spreadsheets/d/1JiQenkAzhKorbw6XY8spdUUtc2-tYYvBVzeMgsmjLwA/edit?usp=sharing

Most cases converged in around 250 iterations, but cases with higher angles of attack or lower Reynolds numbers took longer to converge.
Concluding Remarks: 3 of the runs did not converge properly and resulted in impossible values, so they will need to be removed from the dataset.

# Experiment 7: Mesh for High Fidelity CFD Simulations,

12/30/2025, Ishan Kasam

**Introduction:**

This experiment focused on creating the high-fidelity mesh template that will be used for the smaller set of expensive URANS simulations. The goal was to generate a significantly finer mesh that could better capture vortices and more precise effects while remaining computationally feasible. Since the low-fidelity mesh had already been verified and validated, the purpose of this step was refinement rather than re-validation. The expectation was that a finer mesh using the same modeling strategy would inherit the reliability of the previously validated setup.

- **Methodology**
  I reused the same NACA 4415 airfoil geometry and vortex generator configurations from the medium-fidelity simulations.
    - The airfoil stayed scaled to a 250 mm chord.
    - The same C-shaped domain was kept so results would stay comparable across fidelity levels.
    - The inlet remained 1.25 m upstream, the outlet 5 m downstream, and the far-field boundaries 2.5 m above and below the chord line.
    - Since this domain had already been validated and proven stable, I did not repeat a domain size study. The assumption was that a finer mesh using the same domain would still be physically valid.
- I generated the high-fidelity meshes in ANSYS Fluent using the same poly-hexcore topology to stay consistent with the earlier workflow.
    - The main change was much heavier refinement in the vortex generator wake and far-wake regions.
    - A near-wake refinement box started directly upstream of the vortex generators and extended to 0.1c downstream of the trailing edge to capture shear layers and unsteady vortex formation.
    - A second larger refinement box started at mid-chord and extended 15c downstream and ±1c vertically to capture long wake recovery and transient separation behavior.
- The mesh used much smaller face sizing on the airfoil and vortex generator walls and finer volume cells inside the refinement regions.
    - Inflation layers were also refined compared to the medium-fidelity meshes.
    - The final meshes had about 10.3 million cells for vortex-generator configurations.
    - I did not run a new mesh independence study because this mesh was significantly finer than the validated medium-fidelity meshes, so increasing resolution was assumed to improve accuracy.
- I selected 10 representative operating conditions from the medium-fidelity dataset.

- These were chosen to cover attached flow, near-stall, and deep stall conditions.
- The solver setup stayed nearly the same as before, except the turbulence model was upgraded from steady RANS to URANS k-omega SST so unsteady separation and vortex shedding could be captured.

**Observations and Experimental Data**

The refined meshes showed significantly denser cell concentration in the vortex generator region and extended wake. Cell transitions between refinement zones remained smooth due to the poly-hexcore topology. Estimated simulation time increased to approximately 20–24 hours per case, which was acceptable given the small number of high-fidelity simulations required for the multi-fidelity model.

**Concluding Remarks**

A high-fidelity URANS mesh template was successfully generated using the same validated domain and modeling approach as the medium-fidelity simulations but with significantly increased wake refinement and overall cell count. These simulations will provide the higher-accuracy dataset required for the multi-fidelity machine-learning framework.

# Experiment 8: High-Fidelity URANS Simulation Execution,

# 1/2/2025, Ishan Kasam 

**Introduction**

This experiment documents the manual execution of the high-fidelity CFD simulations after the meshes from Experiment 7 were completed. These simulations represent the upper-fidelity dataset used for the multi-fidelity learning framework. The goal of this experiment was to run transient URANS simulations using the refined ~10.3 million cell meshes and extract higher-accuracy aerodynamic and pressure data for a limited set of representative flight conditions. Because these simulations are computationally expensive, only carefully selected cases from the medium-fidelity dataset were simulated.

**Methodology**

- I chose 10 operating conditions from the medium-fidelity dataset to run at high fidelity.
    - The cases were selected to cover attached flow, near stall, and deep stall so the multi-fidelity model would learn from the most nonlinear regions.
    - Every case was run manually in Fluent first to verify solver behavior before any automation.
- I reused the same geometry, domain, and boundary conditions from the validated medium-fidelity setup.
    - Inlet: 1.25 m upstream of the leading edge.
    - Outlet: 5 m downstream of the trailing edge.
    - Far-field boundaries: 2.5 m above and below the chord line.
    - Boundary conditions: velocity inlet upstream, pressure outlet downstream, symmetry on far-field boundaries, and no-slip walls on the airfoil and vortex generators.
    - Turbulence intensity at inlet and outlet was set to 0.1% to represent low-turbulence freestream conditions.
- The solver was set as pressure-based, transient, and double precision.
    - The turbulence model was upgraded from steady RANS to URANS k-ω SST to capture unsteady vortex shedding and separation.
    - Second-order spatial discretization was used for pressure, momentum, and turbulence equations.
    - Pressure-velocity coupling used the coupled scheme for stability with the refined mesh.
- Each case started with a steady RANS run to create a stable initial flow field.
    - Steady runs continued until residuals dropped below $10^{-5}$ and lift/drag stabilized.
    - The converged steady solution was then used as the starting condition for the transient URANS simulation.
- Transient settings were chosen based on vortex shedding time scales.
    - Time step = 0.0004 s.

    - Total steps = 2000 (0.8 seconds of simulated time).
    - Most cases stabilized around ~1700 steps, but all were run to 2000 for consistency.
- During the transient simulations, I recorded time-history data at every time step.
    - Lift coefficient and drag coefficient.
    - Pressure coefficients at the same 20 chordwise probe locations used in the medium-fidelity runs.
    - Time-averaged values were computed after periodic behavior appeared.
- I monitored every simulation manually to catch instability early.
    - Full case and data files were saved for each run.
    - Time-history output files for aerodynamic coefficients and pressure probes were also saved for later analysis.

**Observations and Experimental Data**

The transient simulations showed small periodic oscillations in lift and drag after the initial transient startup period, indicating vortex shedding and unsteady wake behavior were being captured. Simulation runtime increased significantly compared to medium-fidelity simulations, averaging approximately 20–24 hours per case. No simulations diverged, and all ten cases reached stable periodic behavior before the end of the run.

**Concluding Remarks**

Ten high-fidelity URANS simulations were successfully completed using the refined meshes and transient solver configuration. The simulations captured unsteady aerodynamic behavior that cannot be resolved using steady RANS. These results form the high-fidelity dataset used in the multi-fidelity Gaussian Process regression framework.

# Entry 9: Initial Model Development, 1/12/2025, Ishan Kasam

**Methodology:**

- the goal of this experiment was to build the very first surrogate aerodynamic model that could replace CFD inside the control loop
    - the idea was to predict Cl and Cd directly from AoA, Reynolds number, and VG height so the controller could evaluate thousands of candidate heights quickly
    - at this stage the model used only the medium-fidelity CFD dataset and did not yet include any multi-fidelity correction or URANS data
- the system architecture at this stage was split conceptually into “eyes” and “brain”
    - the neural network estimated AoA from pressure taps (already working)
    - this experiment focused on the first version of the “brain,” which predicts aerodynamic coefficients from the flow state and VG height
- the first kernel choice was the RBF kernel
    - this kernel assumes the function is infinitely smooth and varies gradually everywhere
    - the reasoning at the time was that aerodynamic coefficients should vary smoothly with AoA, Re, and VG height
    - later results showed this assumption was wrong near stall and separation because the physics changes more abruptly than RBF can represent
- the training dataset contained the medium-fidelity CFD results (~150 simulations)
    - inputs: AoA, Reynolds number, VG height
    - outputs: lift coefficient and drag coefficient
    - two completely separate GPR models were trained, one for Cl and one for Cd
    - this separation was intentional because drag is noisier and behaves differently than lift
- the first critical preprocessing step was feature scaling
    - Gaussian processes rely on Euclidean distance in feature space
    - Reynolds number is orders of magnitude larger than AoA and VG height
    - without scaling, Reynolds number would dominate the distance metric and the model would ignore the other variables
    - StandardScaler was used to perform z-score normalization
- the exact standardization procedure was
    - compute the mean of each feature across the dataset
    - compute the standard deviation of each feature
    - transform every input using
        - x_scaled = (x − mean) / standard deviation
    - the scaler was fit only once on the training data and reused for all predictions
- the dataset was split into training and testing sets using an 80/20 split
    - the test set was held back and never shown during training

    - this was done to evaluate whether the model could generalize instead of memorizing the dataset
- the Gaussian Process model was initialized with the kernel
    - ConstantKernel × RBF + WhiteKernel
    - the constant kernel represents overall signal magnitude
    - the RBF kernel represents the smooth aerodynamic response surface
    - the white kernel models noise in CFD outputs
- hyperparameters were optimized automatically during training
    - the optimizer adjusted kernel length scales and noise level
    - this was done by maximizing the log marginal likelihood
    - only two optimizer restarts were used, which later turned out to be insufficient
- after training, predictions were made by computing the covariance between a new query point and the training dataset
    - the prediction is a weighted combination of nearby training samples
    - the model also returns a standard deviation that represents uncertainty
- the model was evaluated using the held-out test set
    - lift model performance
        - $R^2 \approx 0.89$
        - RMSE ≈ 0.11
    - drag model performance
        - $R^2 \approx 0.78$
        - RMSE ≈ 0.006
- the results showed the model worked reasonably in attached-flow regions but performed poorly near stall
    - prediction errors increased sharply for AoA above ~14°
    - the model produced overly smooth predictions and failed to capture the rapid change in drag and lift near separation
- the root cause of the poor performance was identified as kernel choice and dataset limitations
    - the RBF kernel assumes the function is infinitely differentiable
    - real aerodynamic behavior near stall has sharp gradients and regime changes
    - the model also lacked high-fidelity data to correct low-fidelity CFD bias
- the conclusion from this experiment was that the first single-fidelity RBF GPR model was not accurate enough for control
    - the smooth kernel underfit the nonlinear stall region
    - uncertainty estimates were too optimistic in poorly sampled regions
    - this motivated two major improvements for the next iteration
        - switching to a Matérn kernel to allow sharper gradients
        - adding multi-fidelity residual learning to correct CFD bias

# Experiment 10: Initial Autonomous ML + Control Architecture Development, 1/14/2026, Ishan Kasam

**Introduction:**
The goal of this work session was to begin development of a full autonomous control system for deployable vortex generators. The intent was to create a system capable of estimating flight conditions, predicting aerodynamic coefficients, and selecting optimal VG deployment in real time.

**Methodology:**
- I designed the overall architecture as four major subsystems.
- A neural network to estimate AoA and Reynolds number from pressure taps.
- A Gaussian Process model to predict CL and CD from AoA, Re, and VG height.
- A regime-switching control policy to select deployment strategies.
- Safety logic for uncertainty monitoring and envelope protection.
- I began implementing the multi-component architecture in Python.
- Separate modules were created for sensor inversion, prediction, and control.
- Multi-fidelity modeling using residual correction (Delta-GPR) was introduced.

**Observations and Experimental Data:**
The initial architecture grew rapidly to over 800 lines of code and required multiple ML models interacting simultaneously.

**Calculations and Data Analysis:**
This phase was primarily architecture development and did not involve quantitative testing yet.

**Concluding Remarks:**
The system concept was successfully defined. The next step will be testing whether the full architecture is practical and maintainable.

# Experiment 11: Evaluation of Architecture Complexity, 1/18/2026, Ishan Kasam

**Introduction:**
This session focused on evaluating whether the initial architecture was appropriate for trajectory optimization and research deployment.

**Methodology:**

- I tested interactions between the neural network, GPR model, and control logic.
- I reviewed dependencies and deployment requirements.

**Observations and Experimental Data:**
Several major issues were discovered:

- The system had tight coupling between components.
- Changes in one module frequently broke other modules.
- The neural network AoA estimator was unnecessary because AoA is known in flight simulations.
- Multi-fidelity showed only marginal early improvements.

**Calculations and Data Analysis:**
The cost-benefit tradeoff showed high complexity with minimal performance gain.

**Concluding Remarks:**
I decided to abandon the large architecture and extract only the aerodynamic prediction core into a standalone model.

**Experiment 8: Development of Simplified GPR Aerodynamic Model, 1/26/2026, Ishan Kasam**

**Introduction:**
The goal was to build a streamlined GPR model that predicts CL and CD directly from AoA, Reynolds number, and VG height.

**Methodology:**

- I created the simple_gpr.py module.
- Inputs: AoA, Re (scaled to millions), VG height.
- Outputs: CL and CD.
- I standardized inputs using StandardScaler.
- Separate Gaussian Process models were trained for CL and CD using a Matérn kernel.

**Observations and Experimental Data:**
The simplified model was significantly easier to train and maintain compared to the original system.

**Calculations and Data Analysis:**
Initial prediction tests produced realistic aerodynamic values and covered the expected flight envelope.

**Concluding Remarks:**
A stable and maintainable aerodynamic surrogate model was successfully created.

# Experiment 12: Hyperparameter Optimization of GPR Model, 1/24/2026, Ishan Kasam

**Introduction:**
The objective was to improve prediction accuracy by tuning kernel parameters using cross-validation.

**Methodology:**

- I implemented a grid search over kernel parameters.
- 5-fold cross-validation was used and LOOCV.
- Performance metrics included RMSE for CL, CD, and L/D.

**Observations and Experimental Data:**
Hyperparameter tuning produced large improvements in model accuracy.

**Calculations and Data Analysis:**
CL RMSE improved by ~30%.
CD RMSE improved by ~31%.
L/D RMSE improved by ~32%.

**Concluding Remarks:**
Hyperparameter tuning significantly improved performance and validated the importance of systematic model optimization.

# Entry 13: Finalizing Everything In the Model and Documenting all Parameters, 1/25/2026, Ishan Kasam

**Introduction:**

**Methodology:**

- I designed the final model to be a streamlined and safety-aware Gaussian Process Regression system that predicts aerodynamic coefficients from angle of attack, Reynolds number, and vortex generator height.
    - The main goal was to keep the system simple while still physically realistic and reliable enough for flight trajectory optimization and eventual real-world use.
- The workflow starts by loading aerodynamic training data generated from CFD simulations.
    - The dataset spans angle of attack, Reynolds number, and multiple VG heights.
    - During initialization, the model stores the minimum and maximum values of all inputs.
    - These bounds are later used to detect out-of-distribution predictions and warn if the model is used outside the training envelope.
- Reynolds number is rescaled into units of millions before training.
    - This keeps feature magnitudes comparable so the kernel length scales stay physically meaningful.
    - Without scaling, Reynolds number would dominate the feature space and hurt optimization.
- All inputs are standardized using a StandardScaler.
    - Gaussian Processes assume similar feature magnitudes.
    - Standardization improves convergence and hyperparameter tuning.
- I trained two separate Gaussian Process models.
    - One predicts lift coefficient (CL).
    - The other predicts drag coefficient (CD).
    - Both use the same kernel: constant kernel × Matérn kernel (v = 1.5).
    - The Matérn kernel was chosen because aerodynamic behavior is smooth but not perfectly smooth.
    - Hyperparameters were optimized with five restarts to avoid bad local minima.
- Noise is handled using the alpha parameter instead of a WhiteKernel.
    - This avoids double-counting noise and improves uncertainty stability.
- I implemented an optional multi-fidelity mode using the Delta-GPR method.
    - The base model is trained on low-fidelity CFD data.
    - High-fidelity data train a residual model that learns the difference.
    - Final predictions combine base and residual outputs.
    - A spatial gating rule limits the residual correction to regions near high-fidelity data.
- A physical drag floor of CD = 0.004 was enforced.

    - This prevents the optimizer from producing physically impossible drag values.
- The prediction pipeline includes safety checks.
    - Reynolds number must be entered in millions.
    - Warnings are issued if inputs fall outside the training range to avoid silent extrapolation.
- I included a full ISA atmosphere model.
    - This allows air density to be computed from altitude.
    - It enables stall speed estimation and flight regime detection.
- The optimization system calculates required lift from aircraft weight and flight conditions.
    - It sweeps angle of attack to estimate stall speed and CLmax for each VG height.
    - This lets the controller enforce safety margins during optimization.
- The model includes flight regime detection.
    - It uses energy state, altitude, velocity ratio, and angle of attack.
    - A hysteresis mechanism prevents rapid switching between regimes.
- The VG optimization algorithm changes behavior by flight phase.
    - Takeoff prioritizes lift capability.
    - Cruise prioritizes lift-to-drag ratio.
    - Landing balances lift and drag.
    - Safety constraints ensure the aircraft stays below stall limits.
    - Cruise logic biases the system toward retracted VGs unless a clear benefit exists.
- I also implemented an inverse solver.
    - It finds the angle of attack and VG height needed to reach a target lift coefficient while minimizing drag.
    - This is important for flight trajectory optimization.
- Finally, I added an optional Cp inversion pathway.
    - This allows the model to infer flight conditions from pressure sensor data.
    - It enables real-world deployment using onboard measurements.

**Experimental Data and Observations:**

$R^2 = 0.9993$ for Cl and Cd for the just the GPR.

In the flight simulation I developed, it saved 2.55% fuel and improved Cl/Cd ratio by 6.5% when compared against a standard VG height of 0.75mm or x% of the chord.

**Concluding Remarks:**

This finalized model shows great potential to improve efficiency if implemented in the real world, and next steps include trying to improve the accuracy of the flight simulation.

# Project Time Management

| Date | Activity Description | #Hours | Approved |
|---|---|---|---|
| 8/25/2025 | Read article titled *FEM-Aided Modeling and Control of a Tethered Hydrokinetic Energy Kite* and drafted an email for the author, David Olinger | 2.5 | |
| 8/26/2025 | I read an article titled *HPA-MPC: Hybrid Perception-Aware Nonlinear Model Predictive Control for Quadrotors With Suspended Loads* and drafted an email to Guanrui Li of WPI | 2.5 | |
| 8/27/2025 | I read an article titled *Effect of unsteady pressure rise on flame propagation and near-cold-wall ignition*, and I drafted an email for Professor Jayachandran of WPI. | 2.5 | |
| 8/28/2025 | I read an article titled *Minimum-Time Sequential Traversal by a Team of Small Unmanned Aerial Vehicles in an Unknown Environment with Winds*, and I drafted an email for Professor Cowlagi of WPI | 2.5 | |
| 8/29/2025 | I read an article titled, *Micropulsed Plasma Thrusters for Attitude Control of a Low-Earth-Orbiting CubeSat* and I drafted an email for Nikolaos Gatsonis of WPI. | 2.5 | |
| 8/30/2025 | I read an article titled *OriSnake: Design, Fabrication, and Experimental Analysis of a 3-D Origami Snake Robot* and sent an email to the PI of the lab, Cagdas Onal of WPI | 2.5 | |
| 8/31/2025 | I read an article titled *A Shared Autonomous Nursing Robot Assistant with Dynamic Workspace for Versatile Mobile Manipulation*, and I drafted an email for the PI of the lab, Zhi Li of WPI. | 2.5 | |
| 9/1/2025 | I edited, proofread, and improved my emails to send. | 1.5 | |
| 9/12/2025 | Prepared for a call with Professor Berk Calli of WPI. Joined the call with Professor Calli to discuss my project idea as well as possible lab positions. | 2 | |

| | | | |
|---|---|---|---|
| 9/14/2025 | Familiarized myself with basic conceptual knowledge of boundary layer behavior, CFD, and vortex generators | 2 | |
| 9/15/2025 | Learned more about boundary layer behavior, CFD, and vortex generators | .5 | |
| 9/16/2025 | Prepared for and joined a call with Professor Jayachandran of WPI to discuss my project idea about vortex generators and possible lab positions. | 2 | |
| 9/24/2025 | Read the article*: A Shared Autonomous Nursing Robot Assistant with Dynamic Workspace for Versatile Mobile Manipulation* in order to send an email to the author. | 2 | |
| 10/1/2025 | Read the article*: Surrogate modeling for flow simulations using design variable-coded deep learning networks* to understand more about reinforcement learning for airflow prediction and to reach out to the author. | 2 | |
| 10/9/2025 | Read the article*: Dynamic Stall Control Using Deployable Leading-Edge Vortex Generators* to fill the knowledge gap of actuation and sensing methods for deployable vortex generators. | 2 | |
| 10/15/2025 | Read the article: *Closed-Loop Flow Separation Control Using the Deep Q Network over Airfoil* to gain insight as to how machine learning has been used to predict airflow over airfoils, but this has proven inconclusive as it does not use CFD data. | 2 | |
| 10/25/2025 | Read the article*: Review of research on low-profile vortex generators to control boundary-layer separation* to gain understanding of vortex generator geometry parameters and their usual purposes. | 2 | |
| 11/4/2025 | Read the article: *A review of the use of vortex generators for mitigating shock-induced separation.* I read this to understand the different purposes of vortex generators, and the different way their heights can be utilized. | 2 | |

| | | | |
|---|---|---|---|
| 11/8/2025 | Read the article*: Deployable vortex generators for low Reynolds numbers applications powered by cephalopods inspired artificial muscles* to understand current gaps in research for active flow control, especially deployable vortex generators | 2.5 | |
| 11/9/2025 | I tried to create a usable mesh for a simulation of a plain airfoil, but ran into issues with skewness, and aspect ratio because of an abrupt transition between the mesh near the airfoil and the mesh and the free stream area. Ran a sanity-check simulation of a very coarse mesh to verify my software integrity. | 4 | |
| 11/14/2025 | Made a new geometry that can be applied to airfoils of all different VG heights. Bodies of influence were used to better control the transition from the near wall region to the free stream, and that issue was fixed, but a new issue arose with trying to apply inflation layers to the airfoil walls (a crucial step), so nothing was usable. | 4 | |
| 11/15/2025 | Fixed the issue with inflation layers by changing the way that inflation was applied. It was first being applied to the face itself with the boundaries of its edges, but it should have been applied to the body with the faces as boundaries, and optically, it seems like a y+ of 1 was achieved, but it cannot be determined until a simulation is run because it depends on the value of skin friction. | 3 | |
| 11/22/2025 | I tried running the simulation with RANS transition, but there were many issues with it. Many boundary conditions were not properly configured, such as slip conditions on the walls, solver settings, convergence checks, and the mesh was far too fine, leading to convergence issues. | 3 | |
| 11/23/2025 | I read the documentation and then fixed the model selection to SST k-omega with gamma transport equation for transition, but the solver consistently crashed for hours before running any iterations. The culprit was found to be the periodic boundary | 3 | |

| | | | |
|---|---|---|---|
| | condition I was setting which seemed to not be supported by the license version. | | |
| 11/28/2025 | I created a new mesh with different bodies of influence with smoother transitions between boxes, lower element counts, and this mesh could be easily changed to accommodate different airfoil geometries | 2 | |
| 12/7/2025 | I successfully configured 3 geometries of vortex generators of 1.34mm, 2.81mm, 0mm, and I ran simulations of RANS SST k-omega with gamma transport equation for transition. | 8 | |
| **Date** | **Activity Description** | **#hours** | **Approved** |
| 8/25/2025 | Read article titled FEM-Aided Modeling and Control of a Tethered Hydrokinetic Energy Kite and drafted an email for the author, David Olinger | 2.5 | |
| 8/26/2025 | I read an article titled HPA-MPC: Hybrid Perception-Aware Nonlinear Model Predictive Control for Quadrotors With Suspended Loads and drafted an email to Guanrui Li of WPI | 2.5 | |
| 8/27/2025 | I read an article titled Effect of unsteady pressure rise on flame propagation and near-cold-wall ignition, and I drafted an email for Professor Jayachandran of WPI. | 2.5 | |
| 8/28/2025 | I read an article titled Minimum-Time Sequential Traversal by a Team of Small Unmanned Aerial Vehicles in an Unknown Environment with Winds, and I drafted an email for Professor Cowlagi of WPI | 2.5 | |
| 8/29/2025 | I read an article titled, Micropulsed Plasma Thrusters for Attitude Control of a Low-Earth-Orbiting CubeSat and I drafted an email for Nikolaos Gatsonis of WPI. | 2.5 | |
| 8/30/2025 | I read an article titled OriSnake: Design, Fabrication, and Experimental Analysis of a 3-D Origami Snake Robot and sent an email to the PI of the lab, Cagdas Onal of WPI | 2.5 | |

| | | | |
|---|---|---|---|
| 8/31/2025 | I read an article titled A Shared Autonomous Nursing Robot Assistant with Dynamic Workspace for Versatile Mobile Manipulation, and I drafted an email for the PI of the lab, Zhi Li of WPI. | 2.5 | |
| 9/1/2025 | I edited, proofread, and improved my emails to send. | 1.5 | |
| 9/12/2025 | Prepared for a call with Professor Berk Calli of WPI. Joined the call with Professor Calli to discuss my project idea as well as possible lab positions. | 2 | |
| 9/14/2025 | Familiarized myself with basic conceptual knowledge of boundary layer behavior, CFD, and vortex generators | 2 | |
| 9/15/2025 | Learned more about boundary layer behavior, CFD, and vortex generators | .5 | |
| 9/16/2025 | Prepared for and joined a call with Professor Jayachandran of WPI to discuss my project idea about vortex generators and possible lab positions. | 2 | |
| 9/24/2025 | Read the article: A Shared Autonomous Nursing Robot Assistant with Dynamic Workspace for Versatile Mobile Manipulation in order to send an email to the author. | 2 | |
| 10/1/2025 | Read the article: Surrogate modeling for flow simulations using design variable-coded deep learning networks to understand more about reinforcement learning for airflow prediction and to reach out to the author. | 2 | |
| 10/9/2025 | Read the article: Dynamic Stall Control Using Deployable Leading-Edge Vortex Generators to fill the knowledge gap of actuation and sensing methods for deployable vortex generators. | 2 | |
| 10/15/2025 | Read the article: Closed-Loop Flow Separation Control Using the Deep Q Network over Airfoil to gain insight as to how machine learning has been used to predict airflow over airfoils, but this has proven inconclusive as it does not use CFD data. | 2 | |

| | | | |
|---|---|---|---|
| 10/25/2025 | Read the article: Review of research on low-profile vortex generators to control boundary-layer separation to gain understanding of vortex generator geometry parameters and their usual purposes. | 2 | |
| 11/4/2025 | Read the article: A review of the use of vortex generators for mitigating shock-induced separation. I read this to understand the different purposes of vortex generators, and the different way their heights can be utilized. | 2 | |
| 11/8/2025 | Read the article: Deployable vortex generators for low Reynolds numbers applications powered by cephalopods inspired artificial muscles to understand current gaps in research for active flow control, especially deployable vortex generators | 2.5 | |
| 11/9/2025 | I tried to create a usable mesh for a simulation of a plain airfoil, but ran into issues with skewness, and aspect ratio because of an abrupt transition between the mesh near the airfoil and the mesh and the free stream area. Ran a sanity-check simulation of a very coarse mesh to verify my software integrity. | 4 | |
| 11/14/2025 | Made a new geometry that can be applied to airfoils of all different VG heights. Bodies of influence were used to better control the transition from the near wall region to the free stream, and that issue was fixed, but a new issue arose with trying to apply inflation layers to the airfoil walls (a crucial step), so nothing was usable. | 4 | |
| 11/15/2025 | Fixed the issue with inflation layers by changing the way that inflation was applied. It was first being applied to the face itself with the boundaries of its edges, but it should have been applied to the body with the faces as boundaries, and optically, it seems like a y+ of 1 was achieved, but it cannot be determined until a simulation is run because it depends on the value of skin friction. | 3 | |
| 11/22/2025 | I tried running the simulation with RANS transition, but there were many issues with it. Many boundary | 3 | |

| | | | |
|---|---|---|---|
| | conditions were not properly configured, such as slip conditions on the walls, solver settings, convergence checks, and the mesh was far too fine, leading to convergence issues. | | |
| 11/23/2025 | I read the documentation and then fixed the model selection to SST k-omega with gamma transport equation for transition, but the solver consistently crashed for hours before running any iterations. The culprit was found to be the periodic boundary condition I was setting which seemed to not be supported by the license version. | 3 | |
| 11/28/2025 | I created a new mesh with different bodies of influence with smoother transitions between boxes, lower element counts, and this mesh could be easily changed to accommodate different airfoil geometries | 2 | |
| 12/7/2025 | I successfully configured 3 geometries of vortex generators of 1.34mm, 2.81mm, 0mm, and I ran simulations of RANS SST k-omega with gamma transport equation for transition. | 8 | |
| 12/7/2025 | Experiment 1: Effects of Vortex Generators of Different Heights at Re = 1,500,000 | 7 | |
| 12/24/2025 | Experiment 2: Choosing Test Cases for Low Fidelity CFD Dataset | 6 | |
| 12/24/2025 | Experiment 3: Low Fidelity CFD Simulations Mesh and Geometry Creation | 7 | |
| 12/2/2025 | Experiment 4: CFD Validation | 6 | |
| 12/28/2025 | Experiment 5: Practice Low Fidelity CFD Simulations | 6 | |
| 12/29/2025 | Experiment 6: Low Fidelity CFD Simulations | 8 | |
| 12/30/2025 | Experiment 7: Mesh for High Fidelity CFD Simulations | 6 | |
| 1/2/2026 | Experiment 8: High-Fidelity URANS Simulation Execution | 8 | |

| | | | |
|---|---|---|---|
| 1/12/2026 | Entry 9: Initial Model Development | 7 | |
| 1/14/2026 | Experiment 10: Initial Autonomous ML + Control Architecture Development | 6 | |
| 1/18/2026 | Experiment 11: Evaluation of Architecture Complexity | 5 | |
| 1/24/2026 | Experiment 12: Hyperparameter Optimization of GPR Model | 5 | |
| 1/25/2026 | Entry 13: Finalizing Everything In the Model and Documenting all Parameters | 6 | |
| 1/26/2026 | Development of Simplified GPR Aerodynamic Model | 5 | |