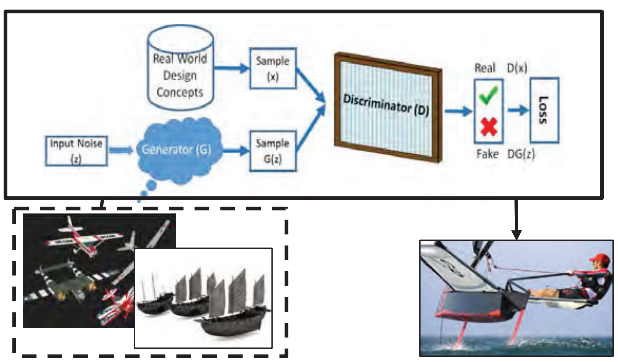
Generative Adversarial Networks for Design Exploration and Refinement

GANDER

Final Presentation*

11 July 2019



THE PENNSYLVANIA STATE UNIVERSITY;
DR. MICHAEL YUKISH (PI)
DR. CONRAD TUCKER
DR. TIMOTHY W. SIMPSON
GARY STUMP
DR. SIMON MILLER
JAMES CUNNINGHAM
DULE SHU

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
30-09-2019	Final	17 Sep 2017 to 30-Sep-2019

4. TITLE AND SUBTITLE
Generative Adversarial Networks for Design Exploration and Refinement (GANDER)

5a. CONTRACT NUMBER
HR0011-18-2-0008

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Dr. Michael Yukish

5d. PROJECT NUMBER
26138

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The Pennsylvania State University Applied Research Labotatory
Office of Sponsored Programs
110 Technology Center Building
University Park, PA 16802-7000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency (DARPA)
ContractsManagementOffice (CMO)
675 North Randolph Street,
Arlington, VA 22203-2114

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER

12. DISTRIBUTION AVAILABILITY STATEMENT
This is approved for public release distribution is unlimited. The recipient agrees that in the release of information relating to this award, such release shall include a statement to the effect that (1) the project or effort depicted was or is sponsored by the Defense dvanced Research Projects Agency, (2) the content of the information does not necessarily reflect the position or the policy of the Government, and (3) no official endorsement should be inferred.

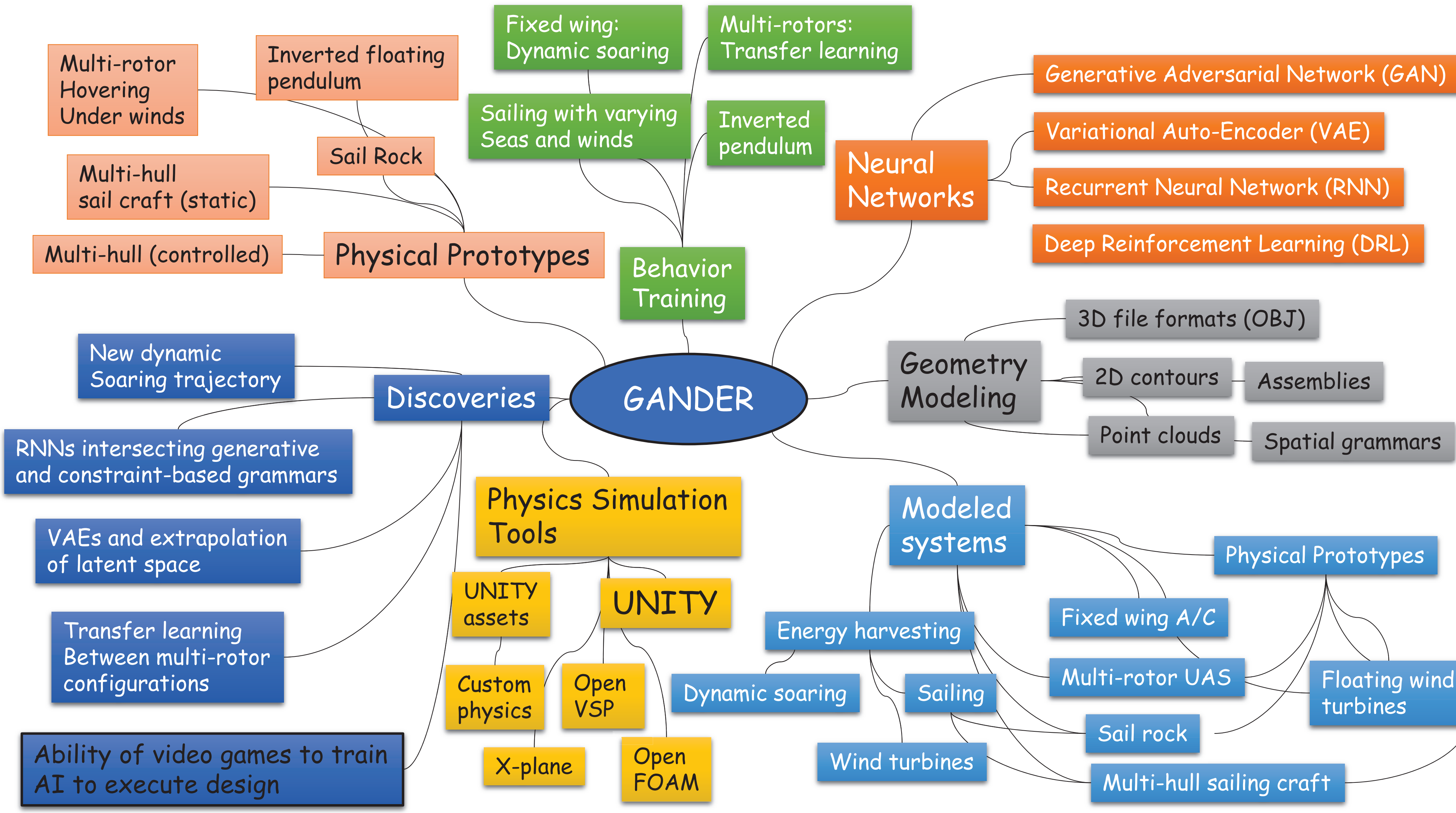
13. SUPPLEMENTARY NOTES

14. ABSTRACT
Penn State developed methods to use artificial intelligence to explore design spaces for complex systems. The methods used game engines to model physics, and a variety of AI architectures (recurrent neural networks, generative adversarial networks) to learn the rules for generating satisfactory designs. The AI learned how to generate both physical configurations and behaviors. The methods were generated on a variety of examples, to include air vehicles, rotorcraft, soaring aircraft, and sailing vessels. Designs were analyzed computationally, and also fabricated and tested via scale models.

15. SUBJECT TERMS

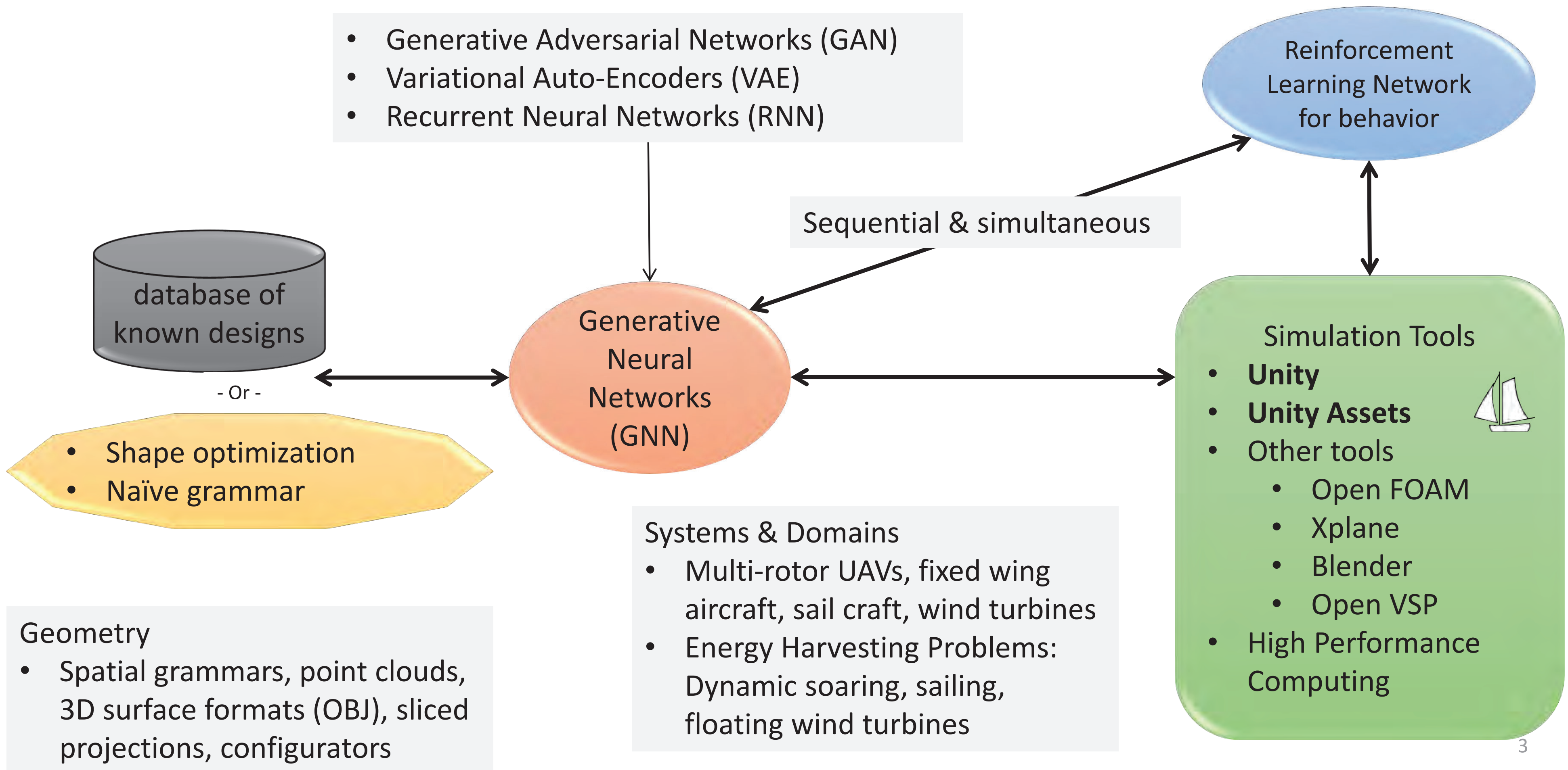
16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
u	u	u	UU	58	Dr. Michael Yukish
					19b. TELEPONE NUMBER (*Include area code*) 814-863-7143

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI-Std Z39-18



GANDER
Physical Prototypes
Multi-rotor Hovering Under winds
Inverted floating pendulum
Sail Rock
Multi-hull sail craft (static)
Multi-hull (controlled)
Behavior Training
Fixed wing: Dynamic soaring
Multi-rotors: Transfer learning
Sailing with varying Seas and winds
Inverted pendulum
Neural Networks
Generative Adversarial Network (GAN)
Variational Auto-Encoder (VAE)
Recurrent Neural Network (RNN)
Deep Reinforcement Learning (DRL)
Geometry Modeling
3D file formats (OBJ)
2D contours
Assemblies
Point clouds
Spatial grammars
Discoveries
New dynamic Soaring trajectory
RNNs intersecting generative and constraint-based grammars
VAEs and extrapolation of latent space
Transfer learning Between multi-rotor configurations
Ability of video games to train AI to execute design
Physics Simulation Tools
UNITY assets
UNITY
Custom physics
Open VSP
X-plane
Open FOAM
Modeled systems
Physical Prototypes
Fixed wing A/C
Energy harvesting
Multi-rotor UAS
Floating wind turbines
Dynamic soaring
Sailing
Sail rock
Wind turbines
Multi-hull sailing craft

Basic Flow of Modeling & Execution



Acquire Training Data

- Generative Adversarial Networks (GAN)
- Variational Auto-Encoders (VAE)
- Recurrent Neural Networks (RNN)

Reinforcement Learning Network for behavior

Sequential & simultaneous

database of designs

- Or -

- Shape optimization
- Naïve spatial grammars
- Etc.

Generative Neural Networks (GNN)

Simulation Tools

- **Unity**
- **Unity Assets**
- Other tools
  - Open FOAM
  - Xplane
  - Blender
  - Open VSP
- High Performance Computing

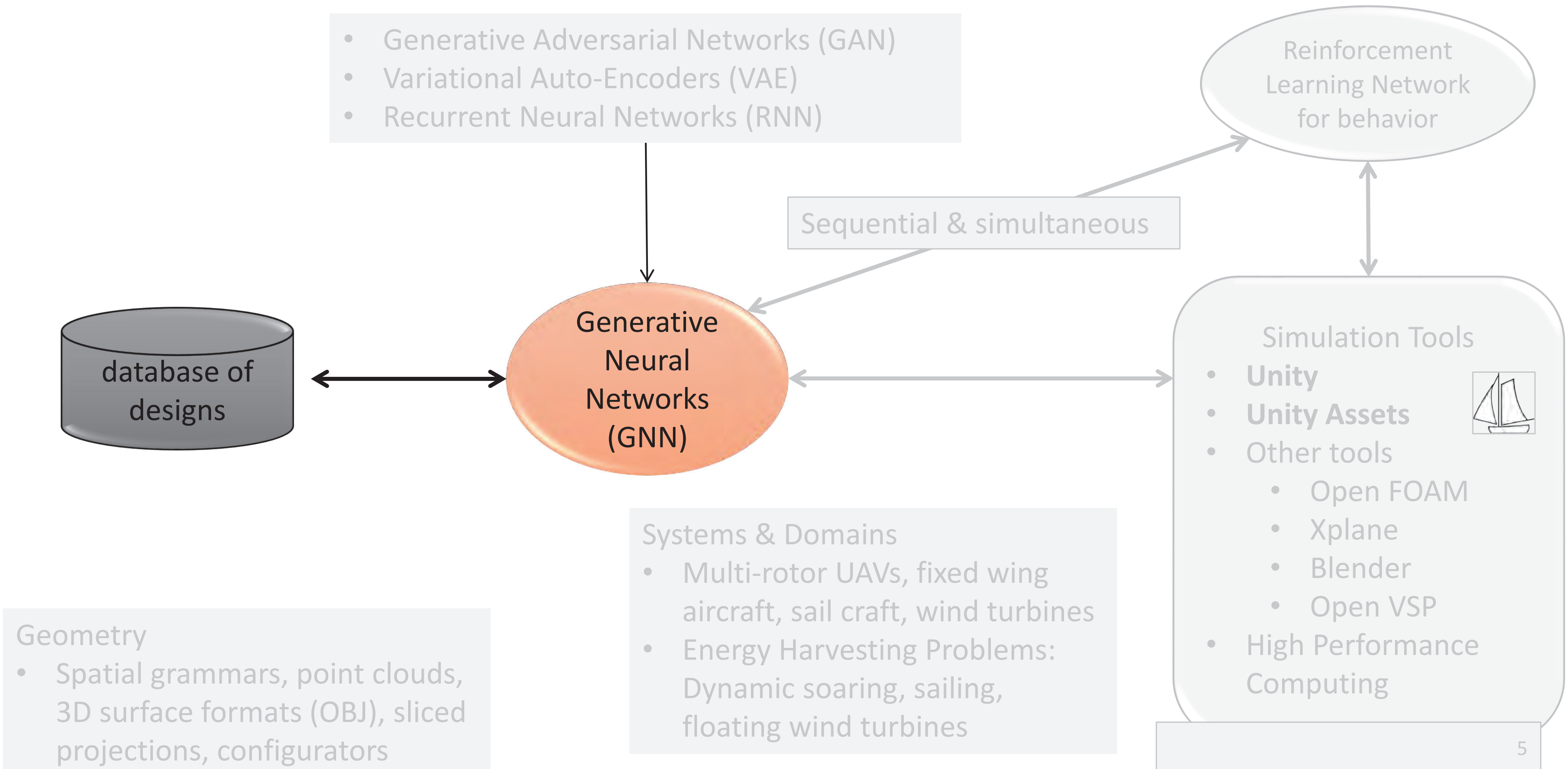
Geometry

- Spatial grammars, point clouds, 3D surface formats (OBJ), sliced projections, configurators

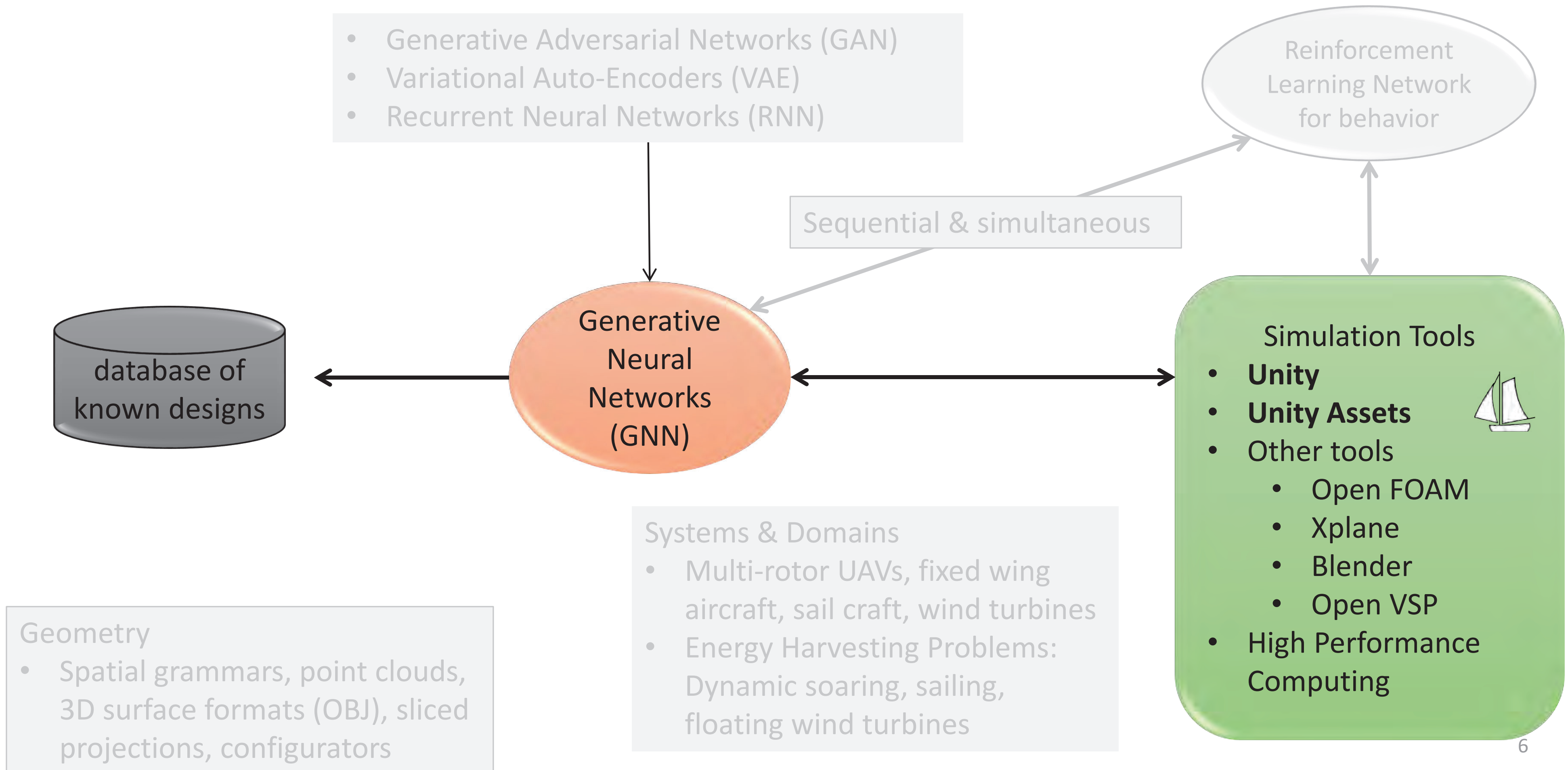
Systems & Domains

- Multi-rotor UAVs, fixed wing aircraft, sail craft, wind turbines
- Energy Harvesting Problems: Dynamic soaring, sailing, floating wind turbines

Bootstrap the GNN (syntax)

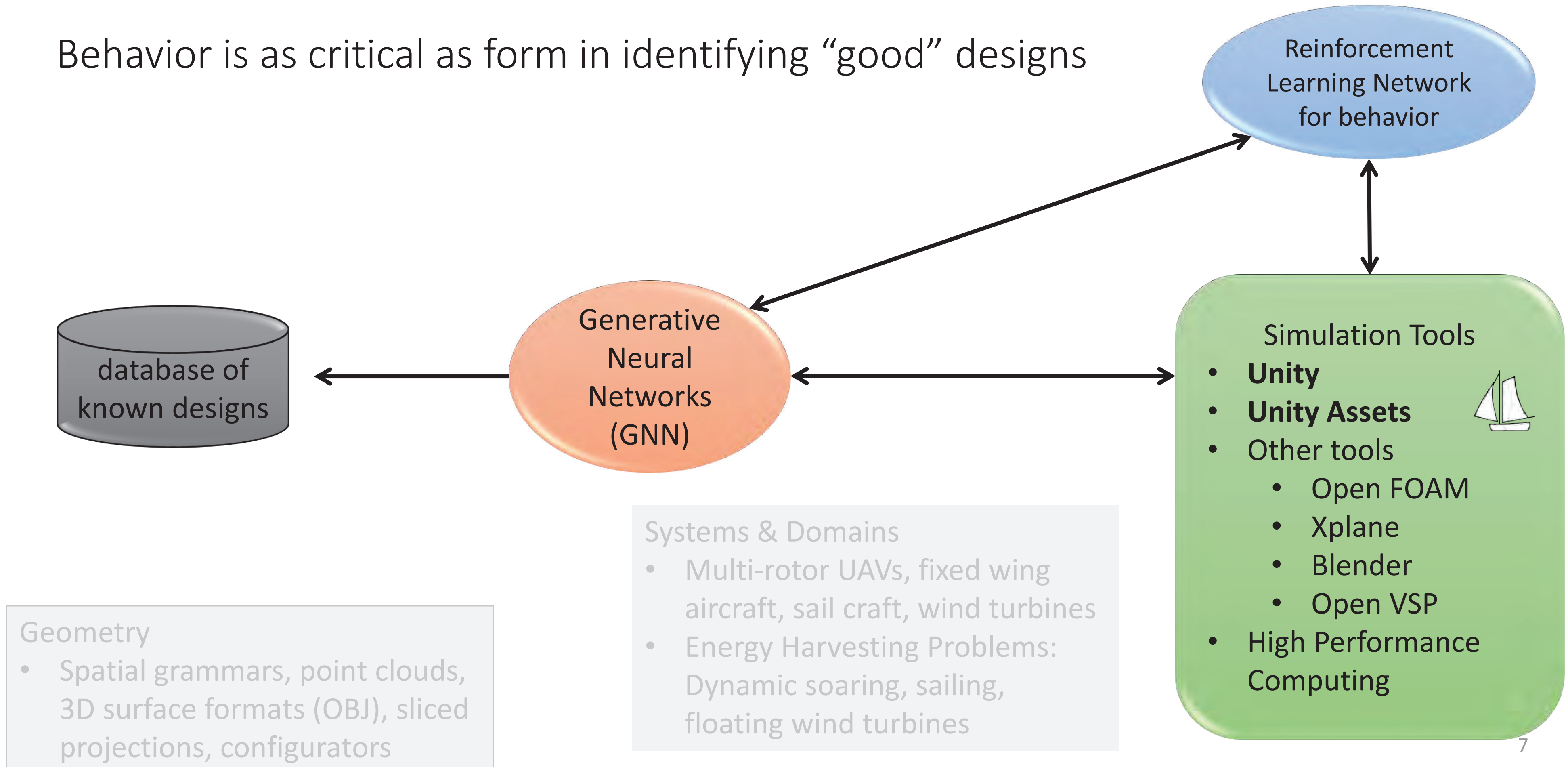


Train Using Simulation (semantics)



Including Behavior in Design Space

Behavior is as critical as form in identifying “good” designs

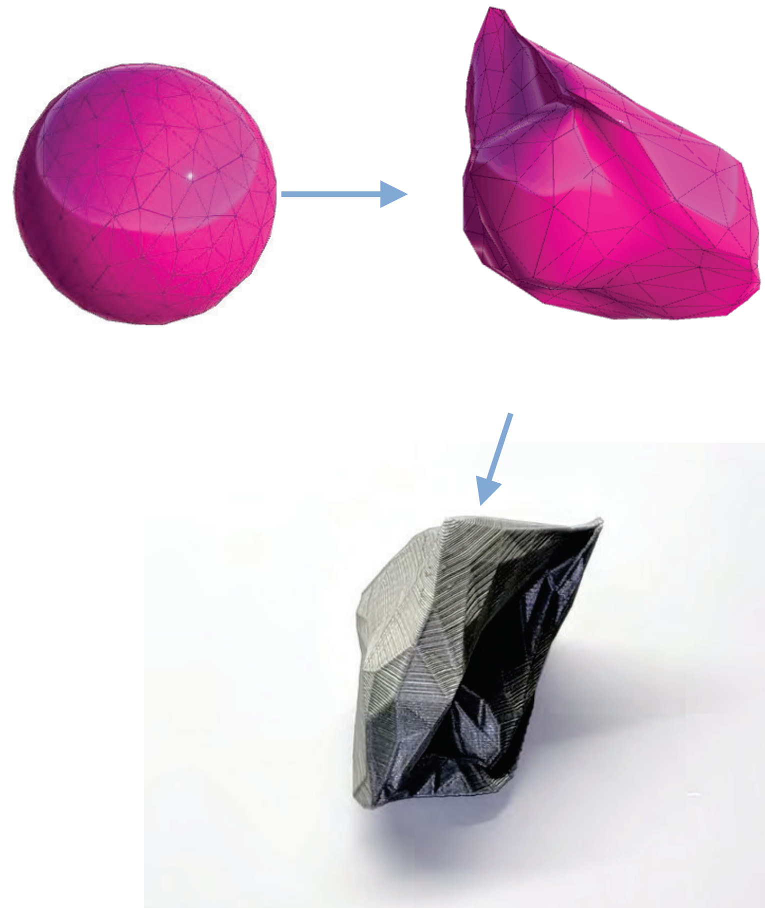


Topics

- Sail-Rock (UNITY to train AI)
- Generative design of valid geometry & learning the underlying physics (syntax & semantics)
- Spatial grammars & watercraft (simultaneous form & behavior)
- Dynamic soaring (RL discovers new trajectories)
- Multi-rotor control (transfer learning of RL)

- *Physical prototyping throughout*

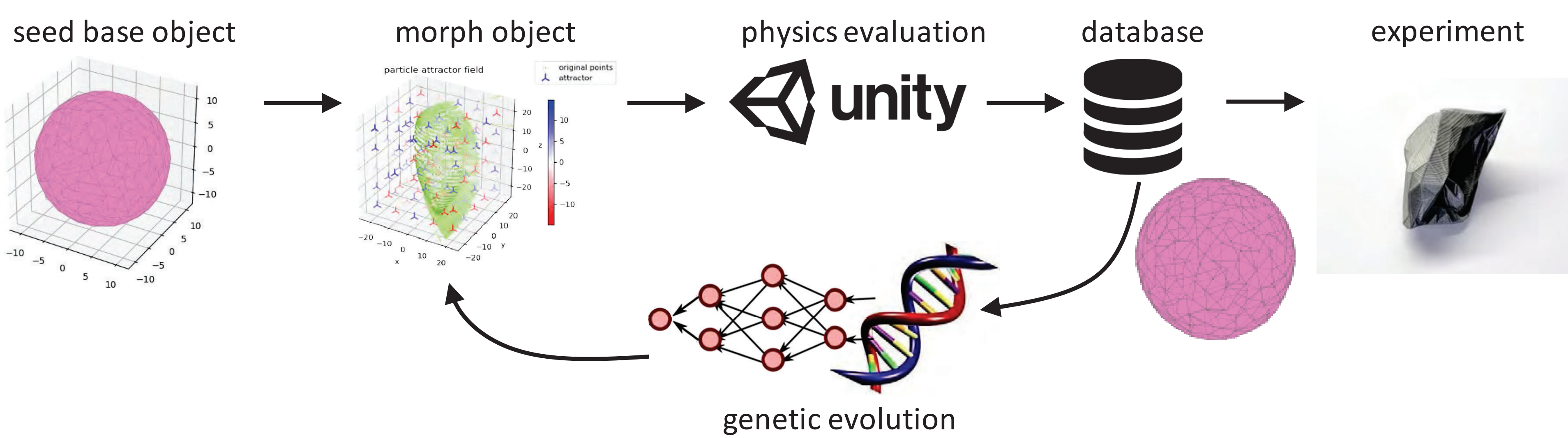
3D Watercraft
Validate using
UNITY for physics



Sail-Rock

First simulation-to-design and fabrication experiment

- Applied shape-morphing strategy to transform a primitive sphere
- Evaluated watercraft in UNITY-based simulation
- 3D Printed best performing design
- Evaluate design in physical experiments
- *Verified project goals*



Unity Game Engine : Strengths

Unity physics capabilities
- Core physics engine
  - Colliders
  - Joints
  - Material Properties (friction, drag)
  - Particle Systems
- Many 3rd party assets from highly active user community
- Readily scriptable for new physics
  - *Provides custom development for additional physics-based capabilities*

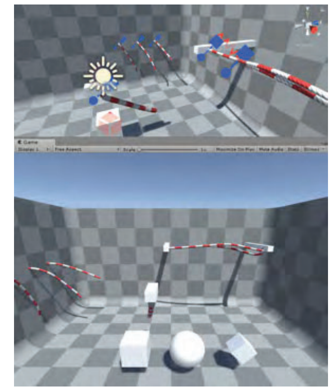
Reinforcement Learning for In-Game Agents

https://www.youtube.com/watch?v=--oTQCysVTs

Unity ML Agent Framework

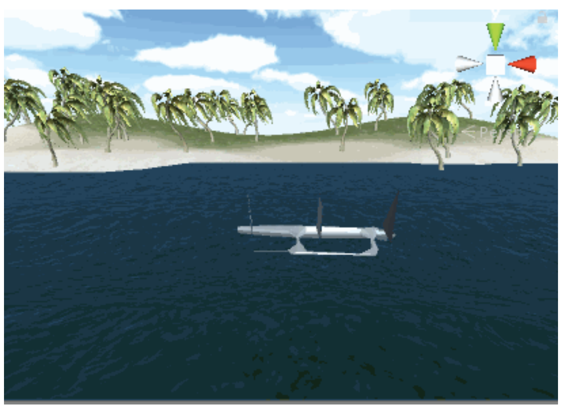
Use Third Party Assets for Rapid Development

https://assetstore.unity.com/



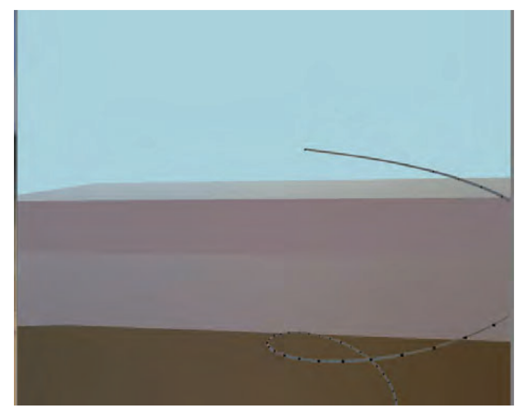
ex: ObiRope : Cable Simulation

custom scripts to simulate aerodynamic and hydrodynamic forces



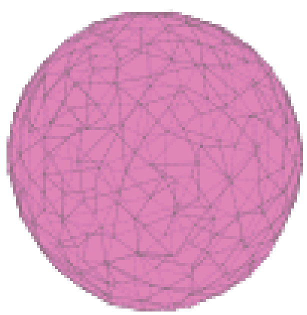
Supports quad copter analysis

Support tether analysis

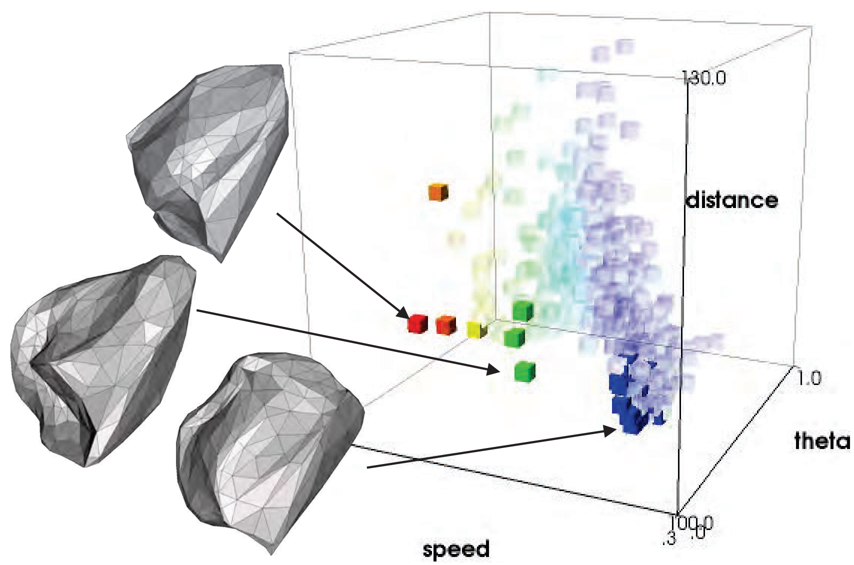


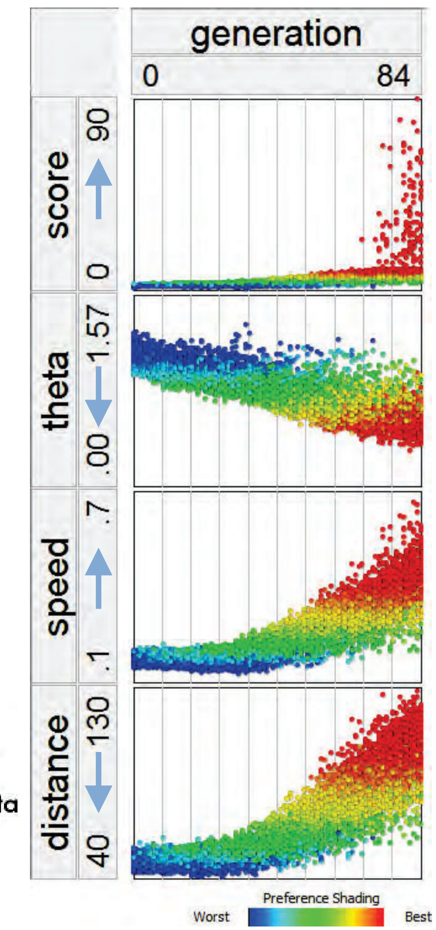
Evolving a Boat from a Sphere

- Multi-objective optimization to design a shape that can sail crosswind quickly
- Genetic evolution of “push/pull” on the “ball of clay” shows emergent sails and keels



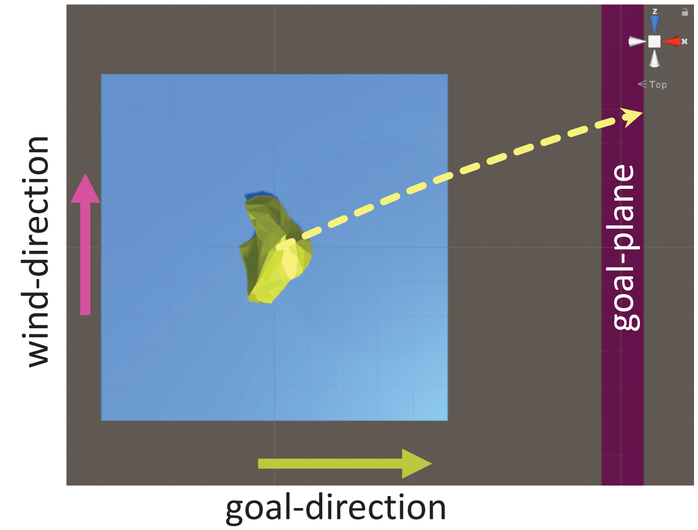
Design Evolution

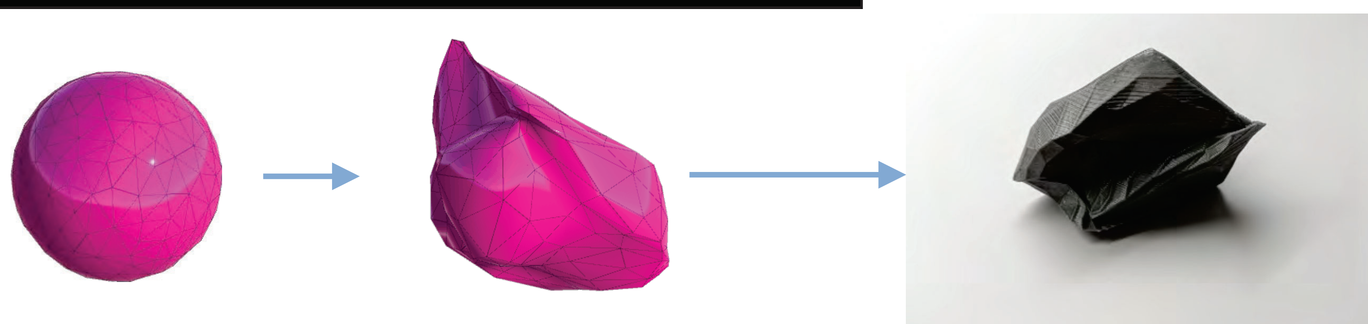




Prototyping the Simulation

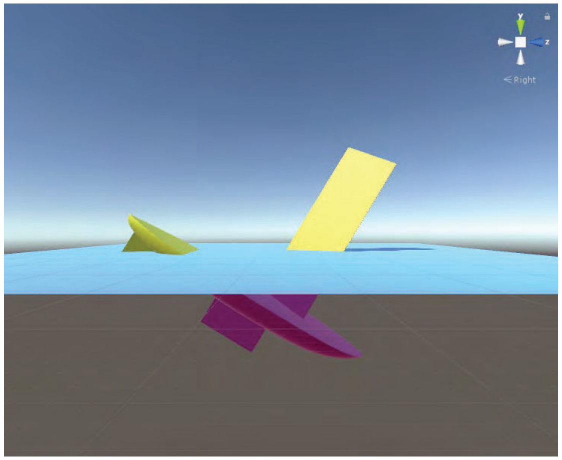
Rapid Physical Testing

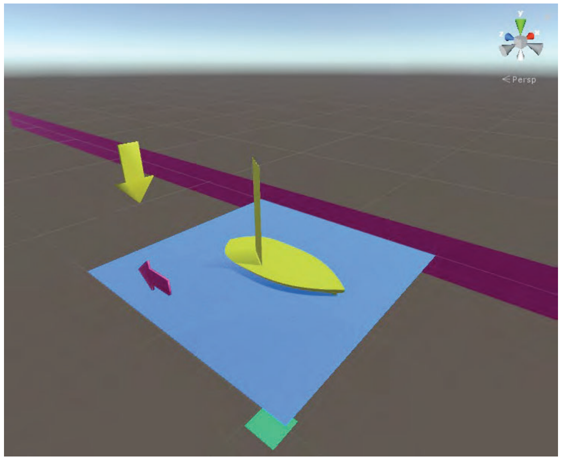




Rapid Prototypes of Design

Fidelity of the Physics





<u>Above the Water</u>

- Air resistance from motion over mesh
- Wind loading is treated as momentum transfer applied orthogonal to the mesh “facing” the wind

<u>Below the Water</u>

- Buoyancy computes split-mesh triangular-prism volumes
- Viscous resistance computes mesh-level drag
- Pressure drag uses empirical quadratic relation
- Slamming forces uses the objects acceleration and swept water volume

<u>Electro- & Magneto-statics</u>

- Lorentz force where the body has a known charge

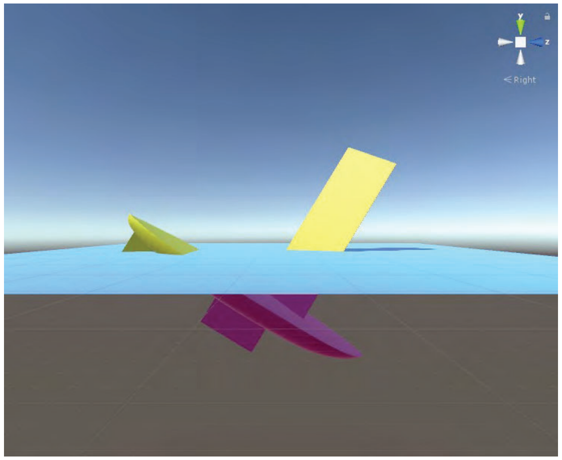
Adding new Physics is a simple process of writing a little **C#** code and enabling the computation into the physics update

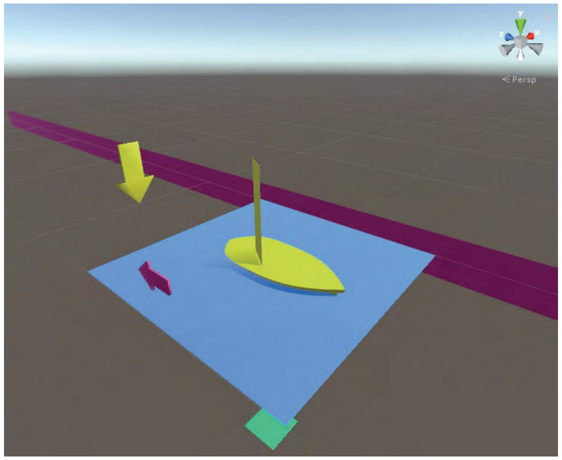
- e.g., *the EM took less than 2days to implement, test, and deploy*

Impact

- Proved game engines are suitable for conceptual design
  - Demonstrated multi-physics, and creating and adding new physics
  - Inexpensive (free for academia!)
  - Can spread across cores, in batch
  - Incorporate reinforcement learning (used later)
  - Generates valid results as confirmed by experiments

- Surprising design result validated by experiment!
  - Efficacy of *Sail-Rock* was unexpected



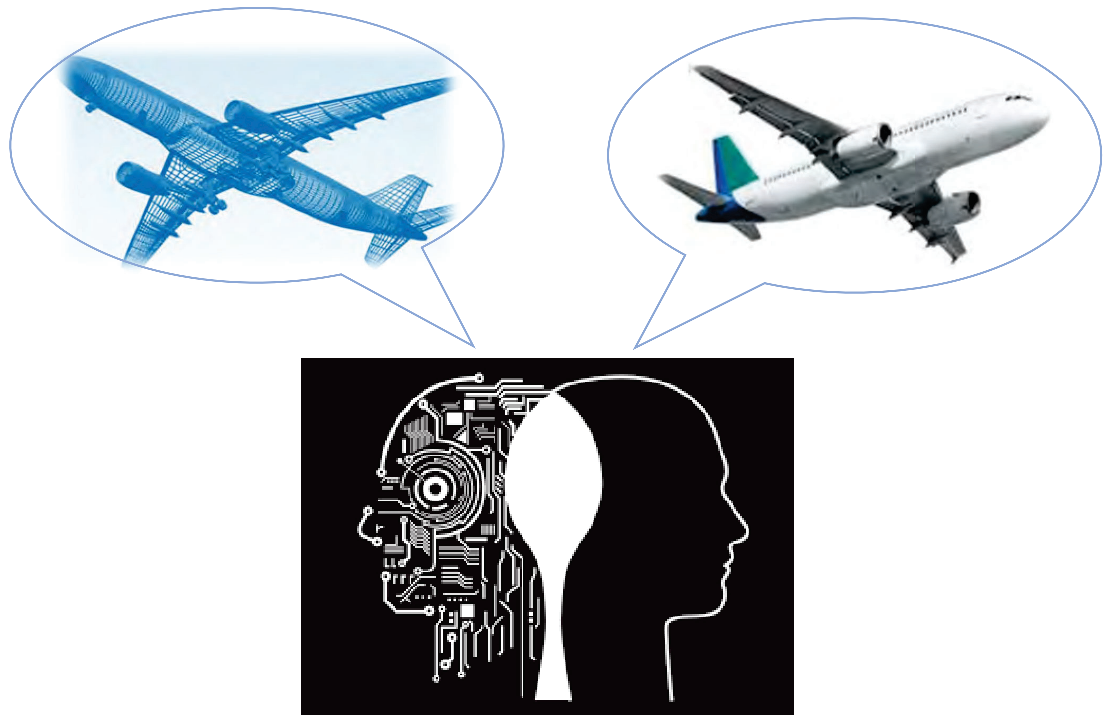


Generative Design using AI

Objective:

- Train a neural network to be an expert 3D modeler and be able to explore (and generate surprise!) in the design domain

- The neural network modeler takes both the form and the function into consideration in its design process



BLUF: successfully trained Neural Networks to organize 3D Points into geometries of interest

Creates valid geometries that are manifold

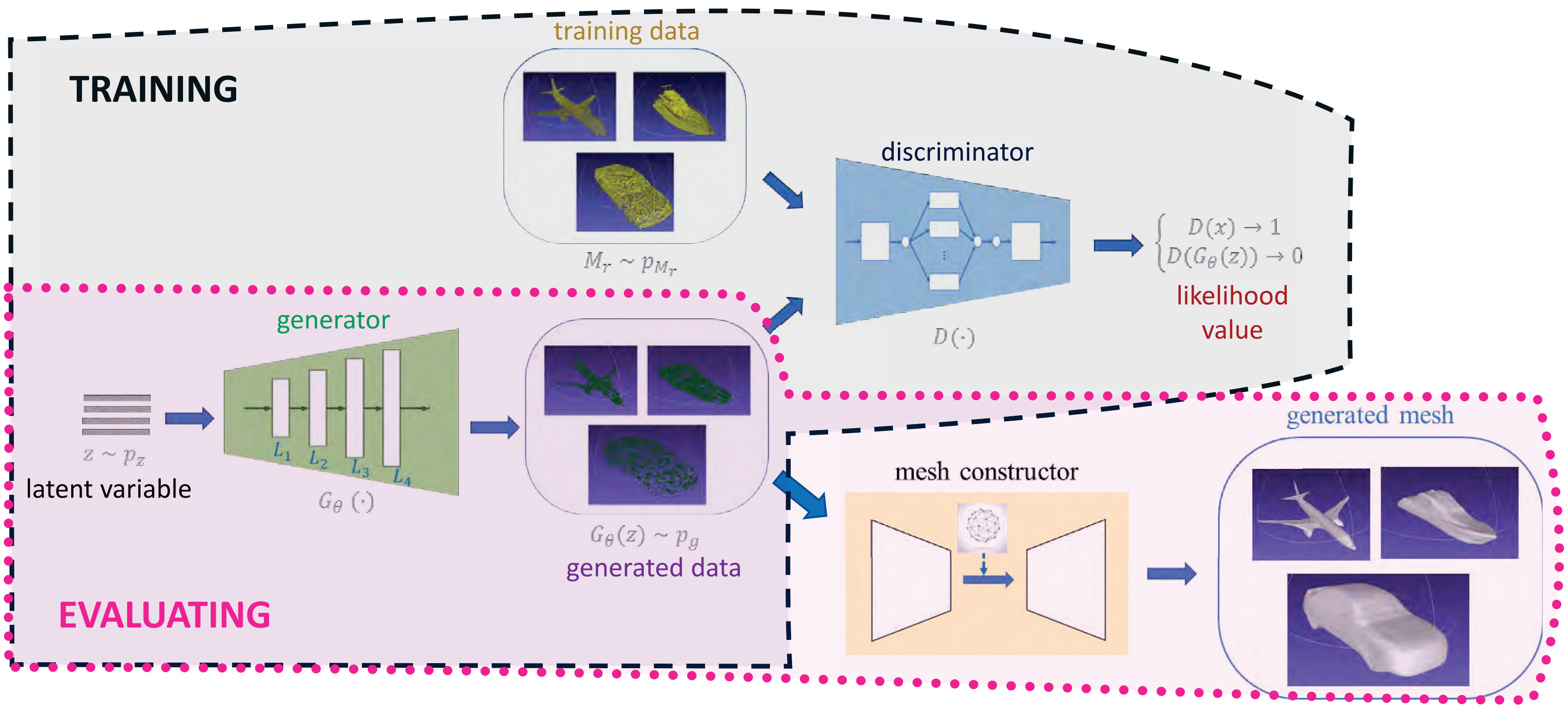
Learns the relationship between form and function

Bridge *classes* of objects to create variety (sea/air/land)

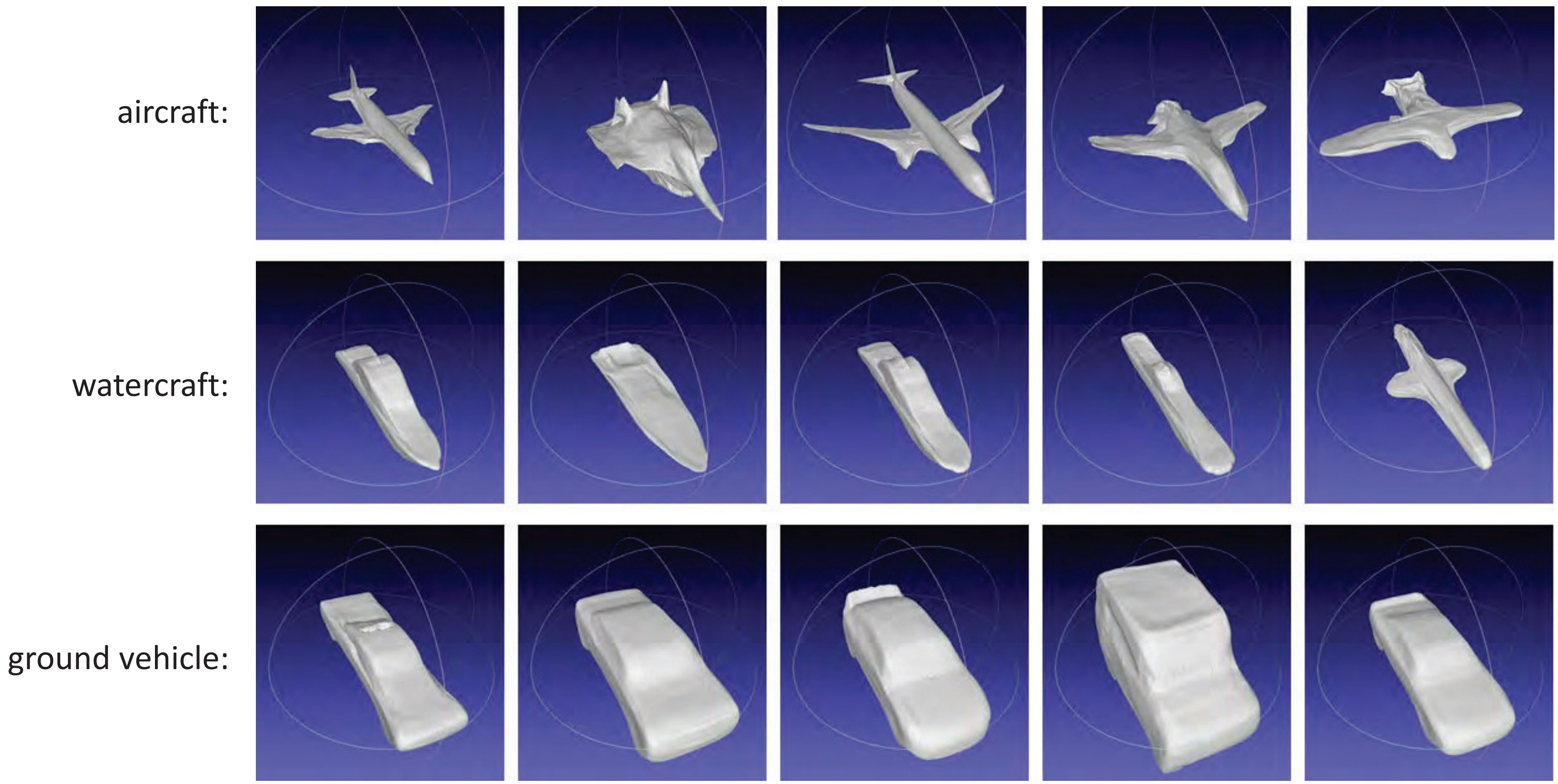
http://www.engr.psu.edu/datalab/

Generative Design with Multiple Classes of 3D Objects

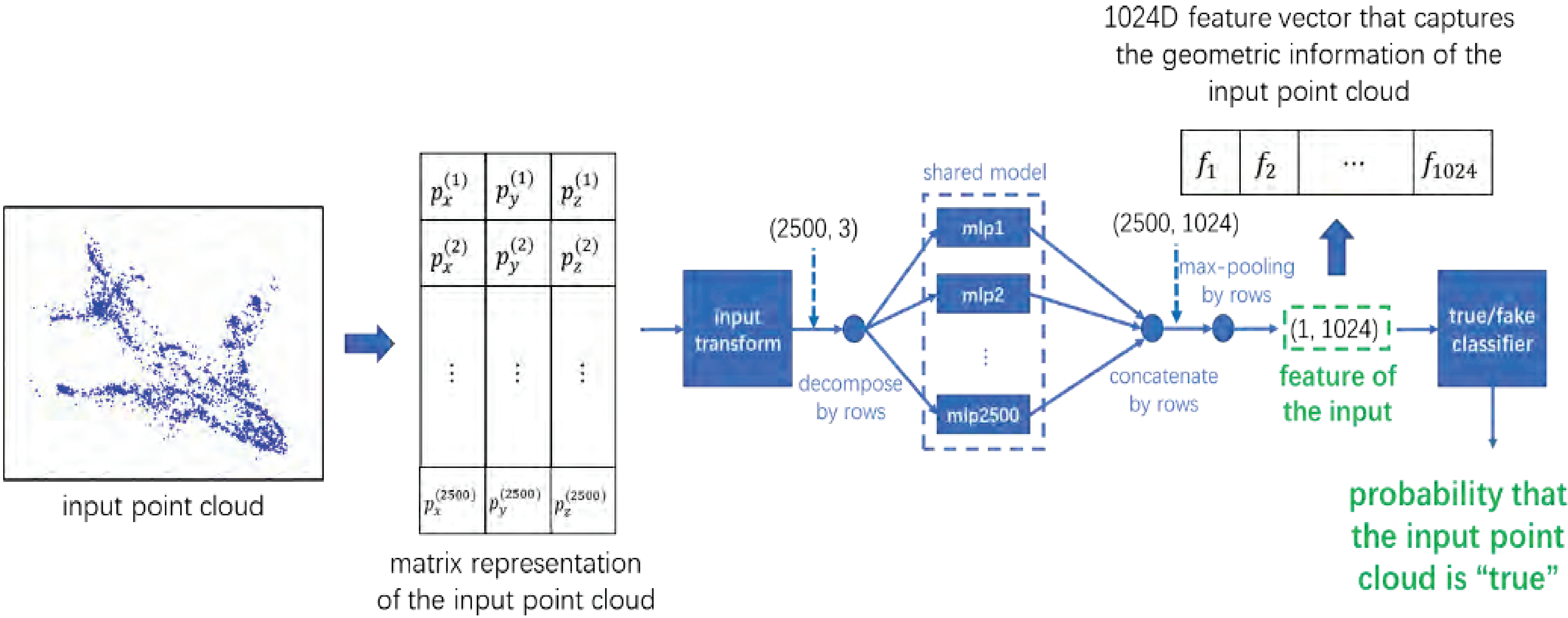
- Training a GAN on a variety of classes of 3D objects, the GAN can learn to generate a wide variety of designs and interpolate between them
- Use a Generative Adversarial Network to learn how to generate designs that are *not incorrect*



Examples of multi-class generated designs

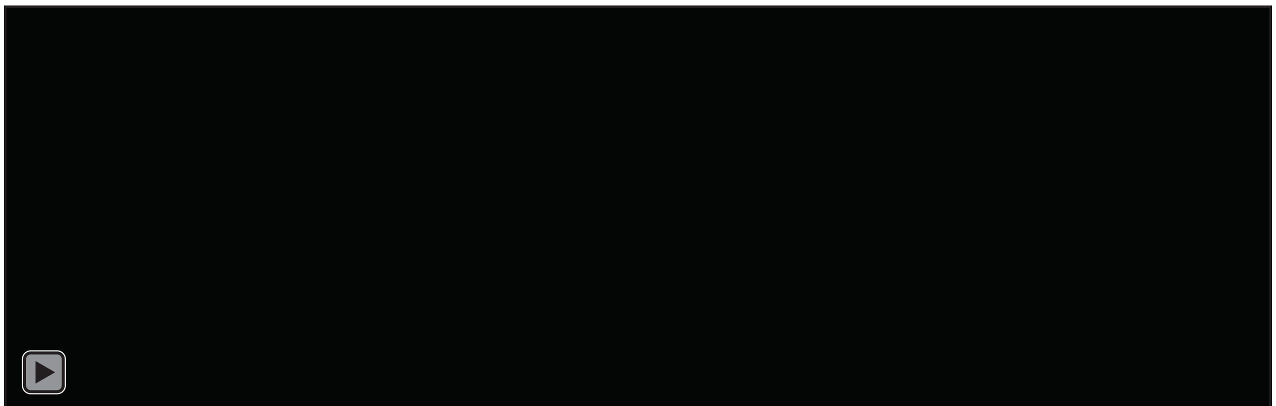


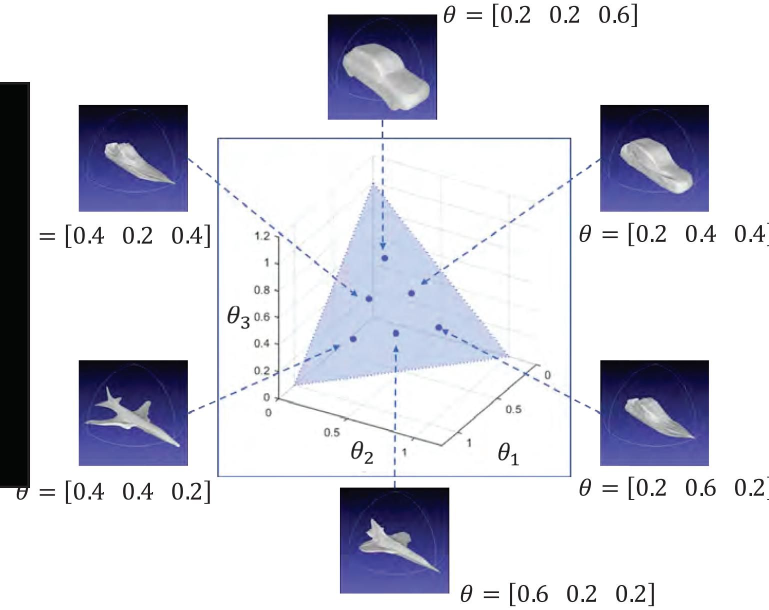
Approach: Generative Adversarial Networks (GANs) create arbitrary geometries that can be combined to form novel shapes



Can Mix Different *Classes* of Designs to Create New Designs

- Design transformation via linear combination of three latent variables
- Mixtures of air, land, & sea



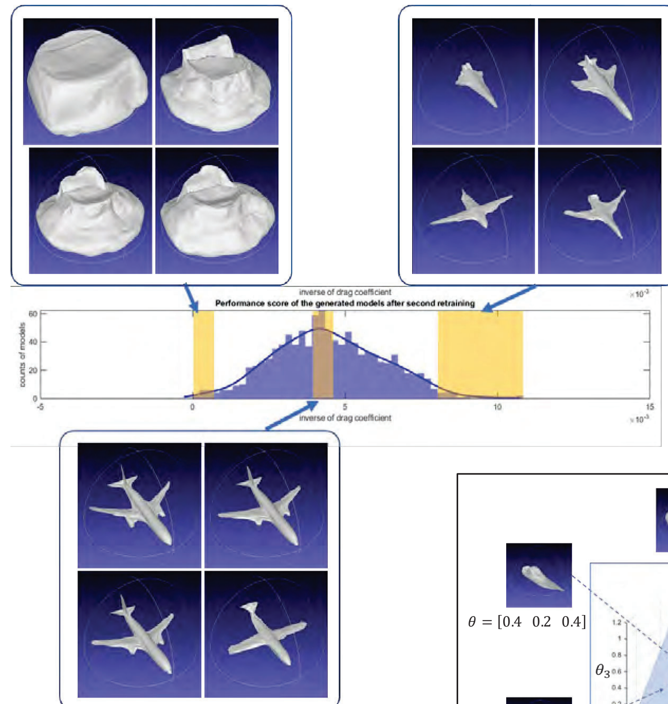


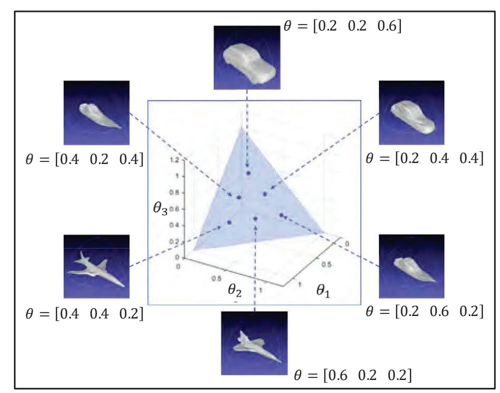
transformation of design derived from z_θ

variation of θ

Accomplishments & Impacts

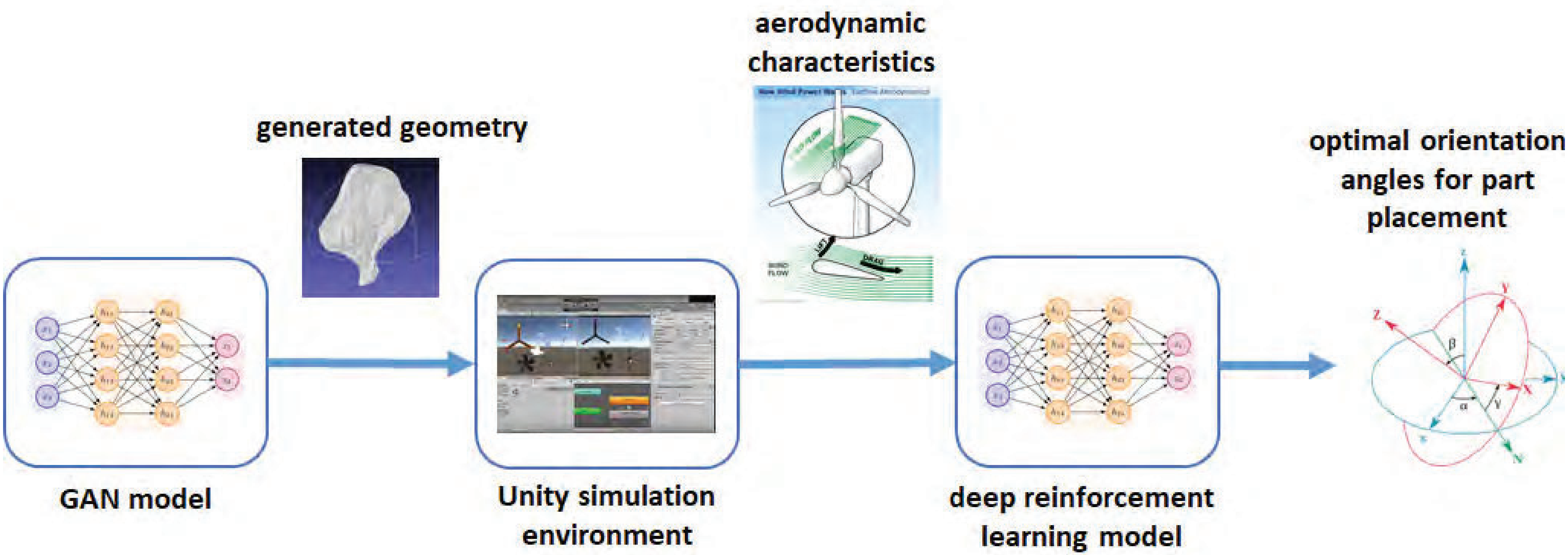
- Single trained GAN can generate a wide range of geometries
    - Geometries meshable, enabling analysis
- Can interpolate between disparate classes
    - Air, land, and sea
- Can *extrapolate* beyond the convex hull of the design space
    - Control direction
    - **Generate *surprise***



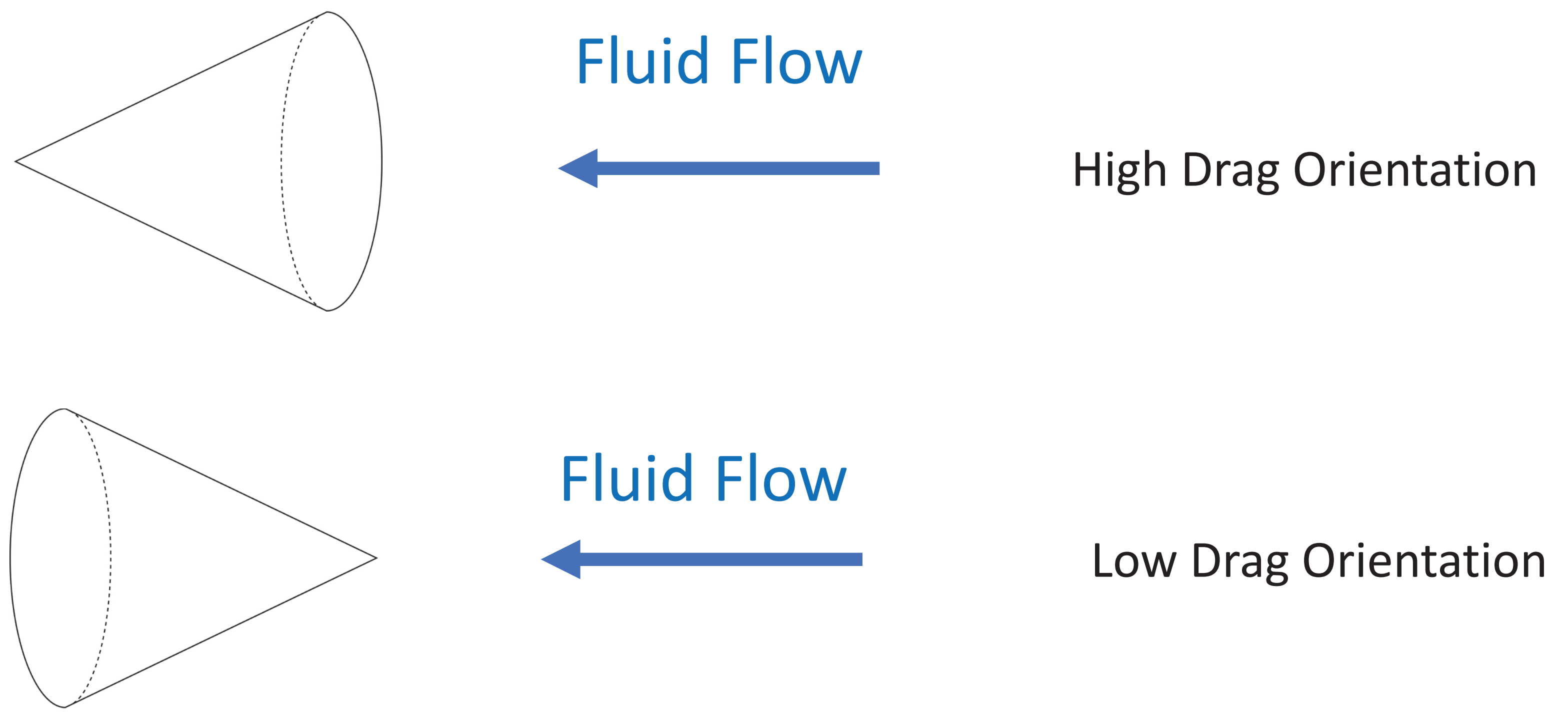


AI to Learn Physics

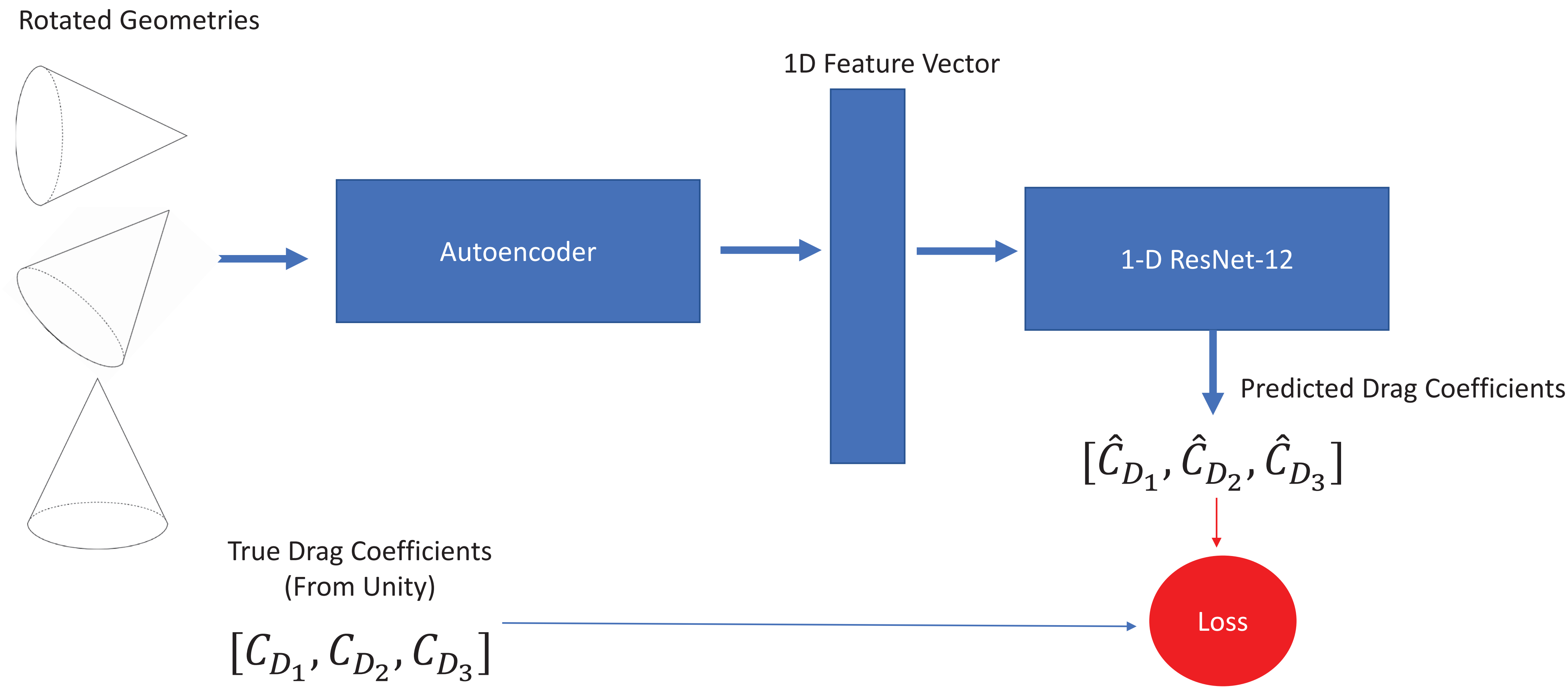
Accomplishment: demonstrated NN learning relationship between shape and physics (e.g., drag)



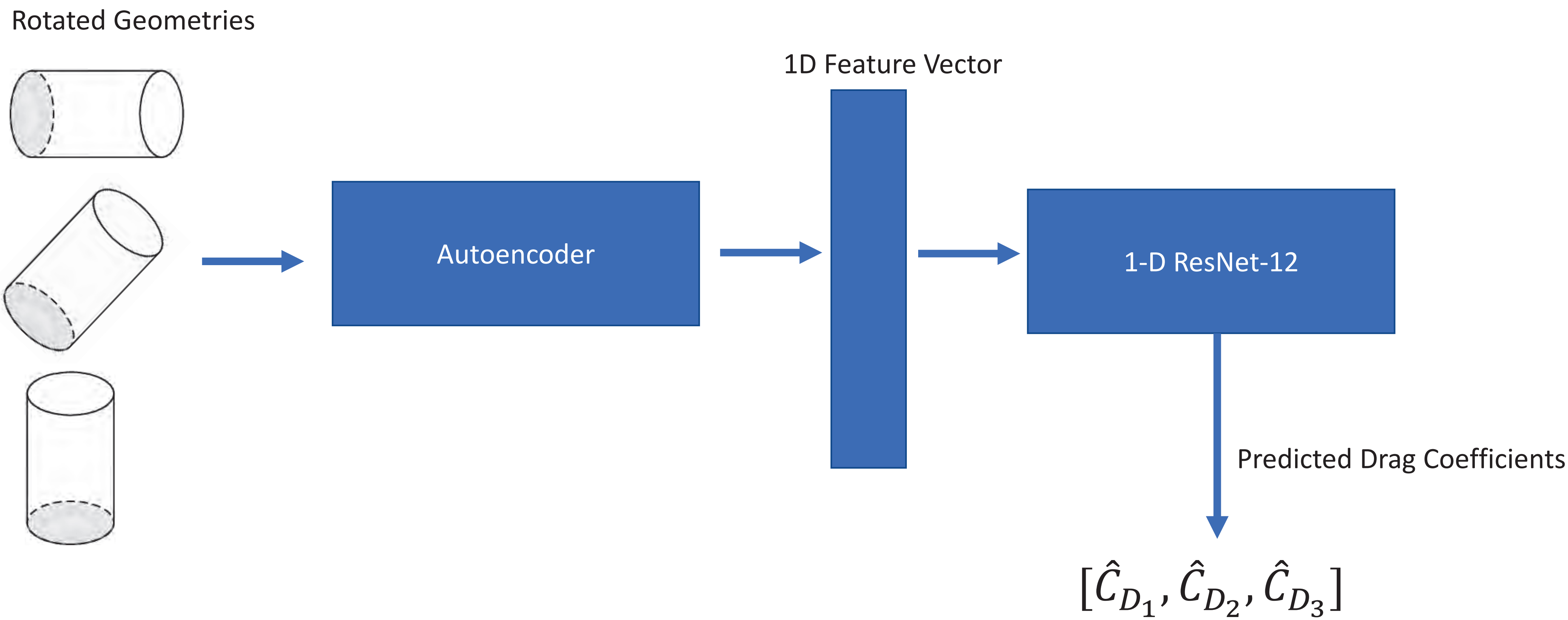
Example Demonstrating Breakthrough: Learning Drag-reducing Orientations of 3D Geometries



Training stage: learning drag coefficient versus orientation from database of geometries

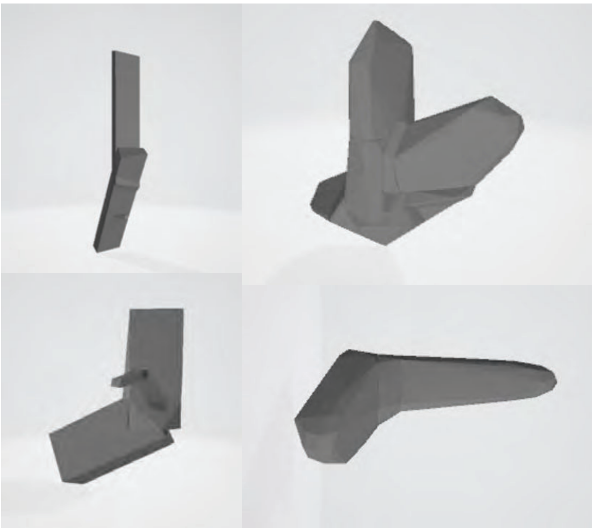


Testing stage: evaluating predictions on new geometries & orientations

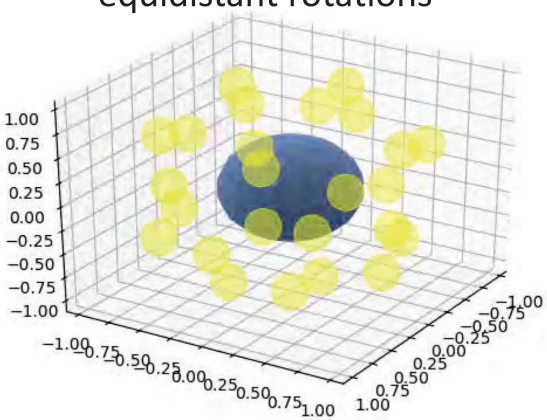


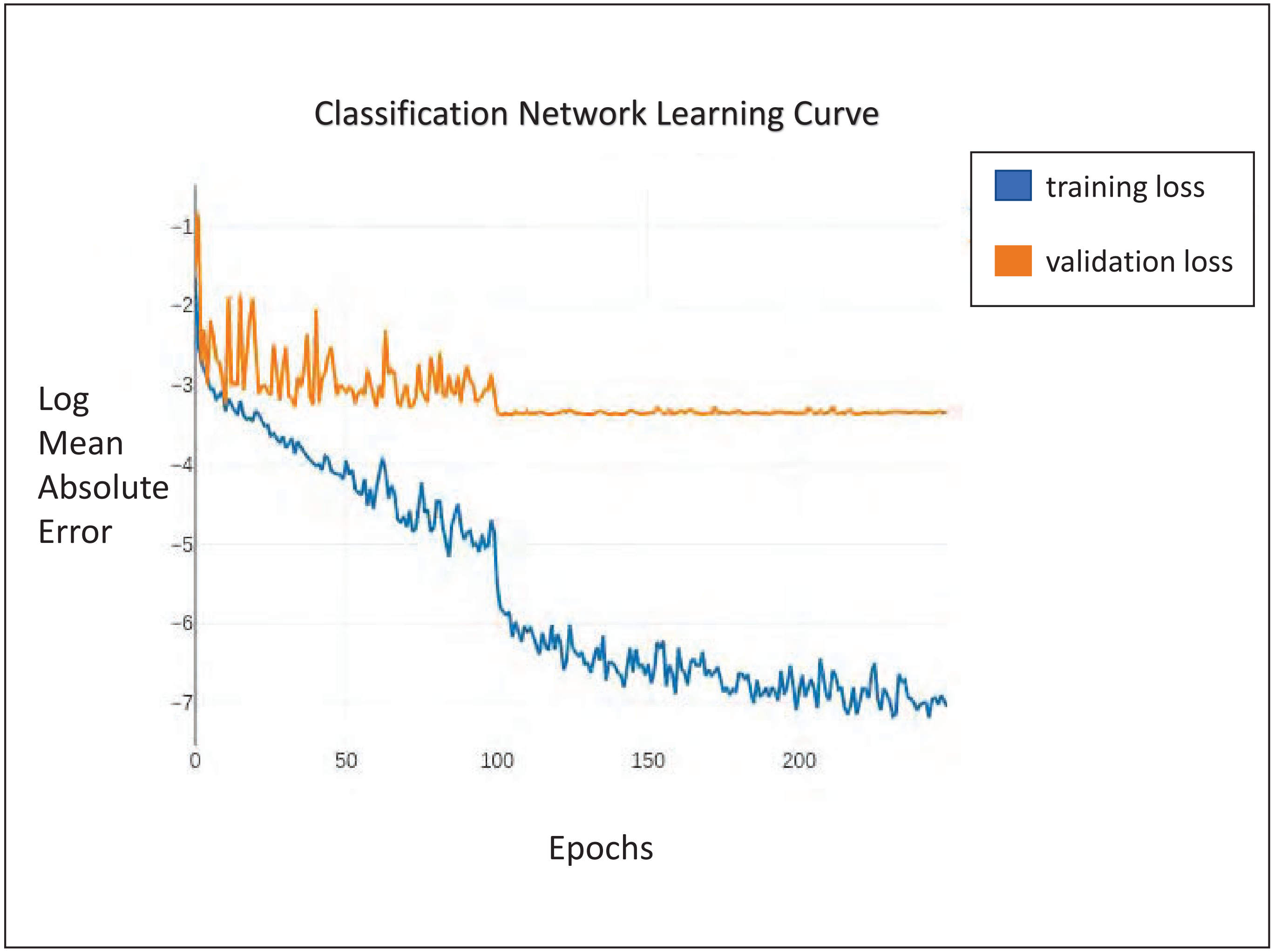
Performance Results/Metrics

Training set of 1000 Geometries



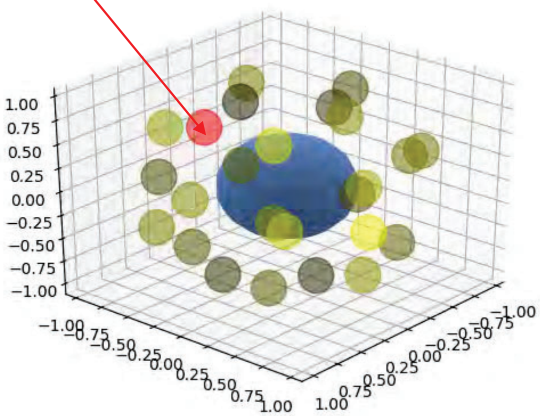
Evaluate at 25 spherically equidistant rotations





Performance Results/Metrics

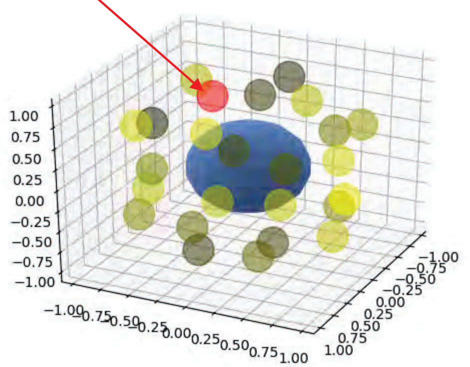
Predicted
optimal
orientation



Matches True Optimal Orientation

Example Predictions

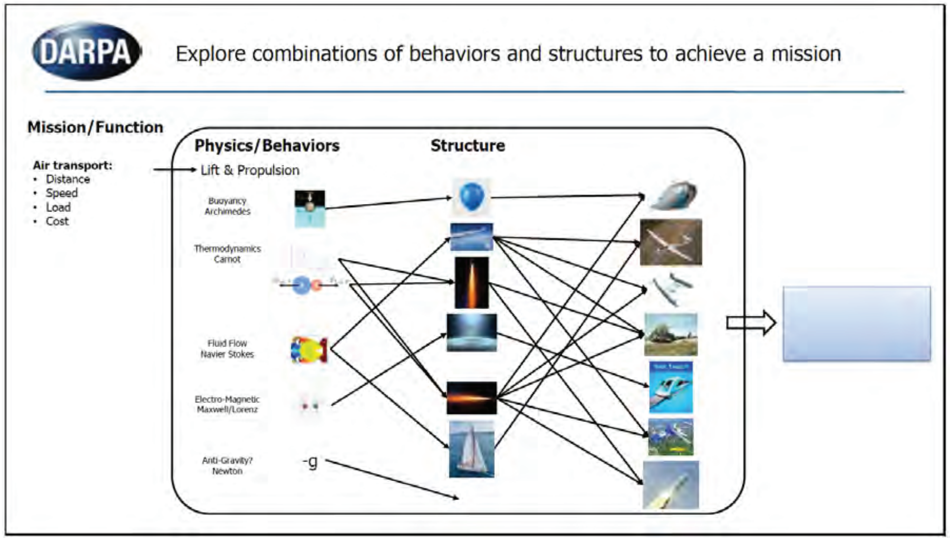
Predicted optimal orientation



Matches True Optimal Orientation

Accomplishments & Impacts

- Demonstrated AI internalizing the knowledge of a designer
  - In previous effort, demonstrated AI creating valid shapes (syntax)
  - Now demonstrated AI can arbitrarily attach physical phenomena to the shapes (semantics)



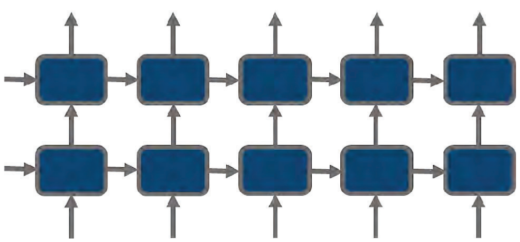
- ***Impact**: can use AI for design, at vastly greater scales of application*
  - *Millions of AI versus tens of designers*

Grammar-Based Design

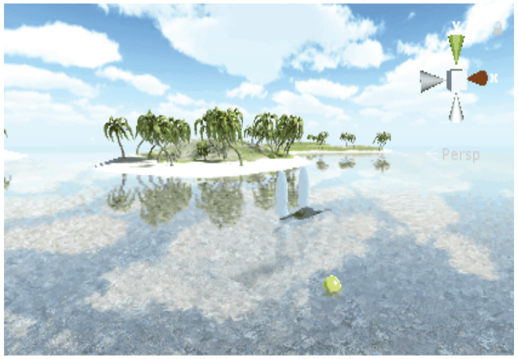
Spatial Grammars & Sail Craft

Fun Design Goal : Give the designer the ability to evolve and adapt designs rapidly in response to changing requirements and provide a thorough understanding of trade-offs early in the design process.

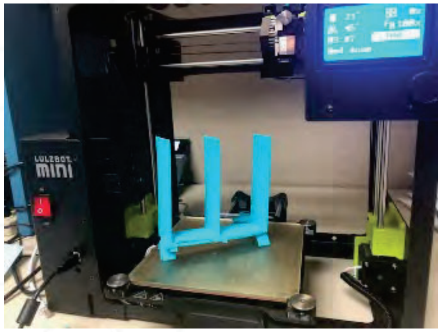
Spatial Grammars and RNN : Spatial grammars define assembly designs (as strings) and RNNs learn the language of good designs



Game Engines : Evaluate Designs with Reinforcement Learning

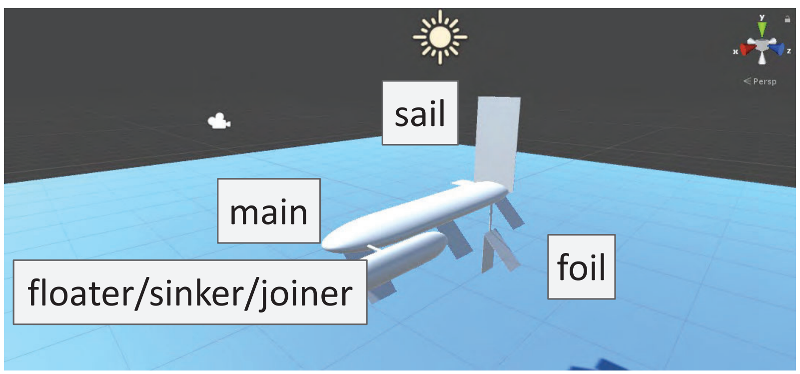


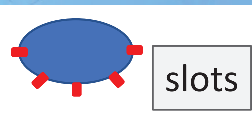
Additive Manufacturing : 3D Polymer prototypes



Generative Grammar for Boat Assembly

- Defining boat assemblies as strings to train a charRNN
- AI *learns the language* of design and assembly as
  - Production Rules G
  - Geometric Constraints C





- Using a context-free grammar to define the components and connections of a boat assembly

Rules in BNF format:

```
start : “main” open sail sail sail close
                open
                open slot slot slot slot slot close
                open slot slot slot slot slot close
                open slot slot slot slot slot close
                close

slot : “foil” | “empty” | sa

sa:  open strut  nodal close

nodal : node open slot slot slot slot slot close

node : “floater” | “sinker” | “joiner”

strut : “foil” | “rod”

sail : “big” | “small” | “none”

open : ‘{‘

close : ‘}’
```

Example of Validated String

main { small none none } { { empty foil { rod joiner { empty foil foil empty empty } } empty foil } { empty foil empty foil empty } { { rod sinker { empty empty empty empty empty } } foil empty empty empty } }

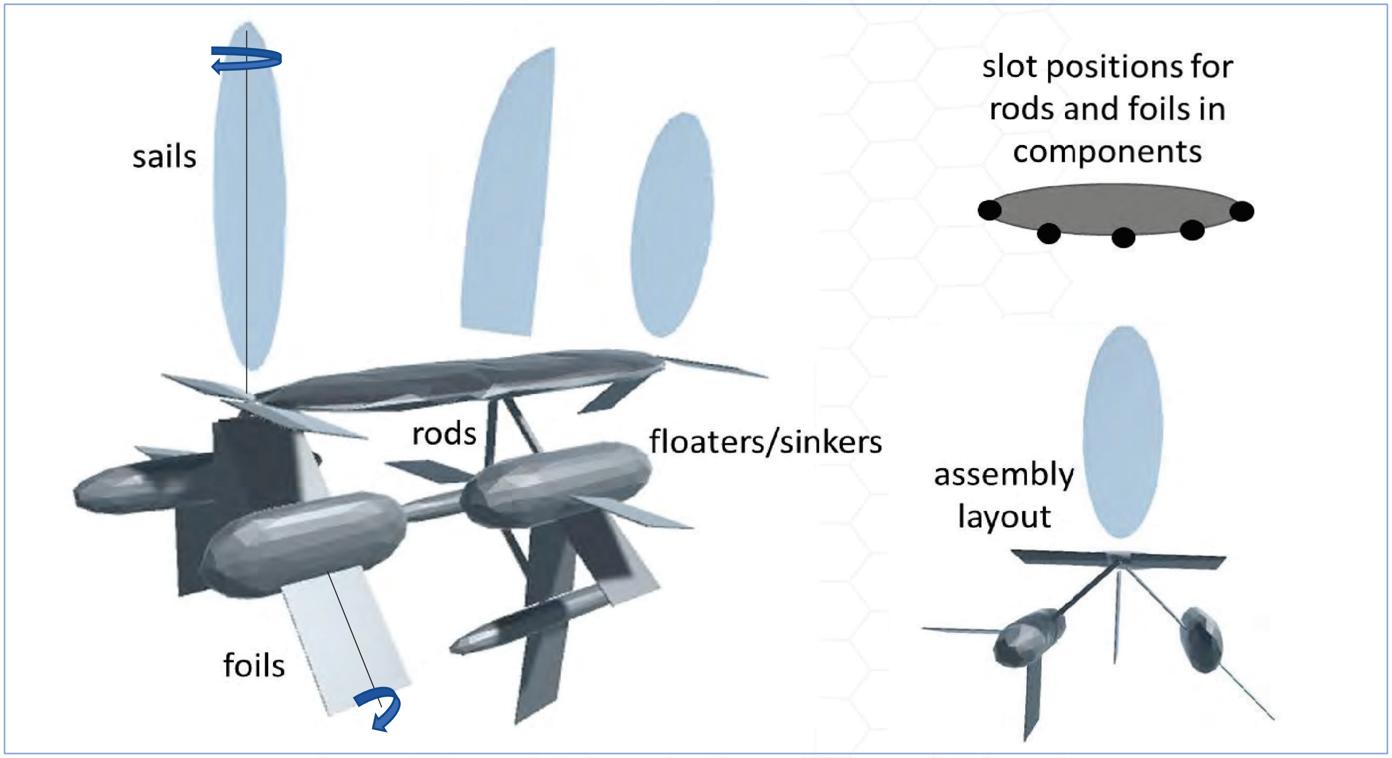
PennState
Applied Research Laboratory

Grammar 2.0 : Specify Arbitrary #s Control Surfaces

Spatial Grammar : a set of rules to that an algorithm can use to generate example assemblies

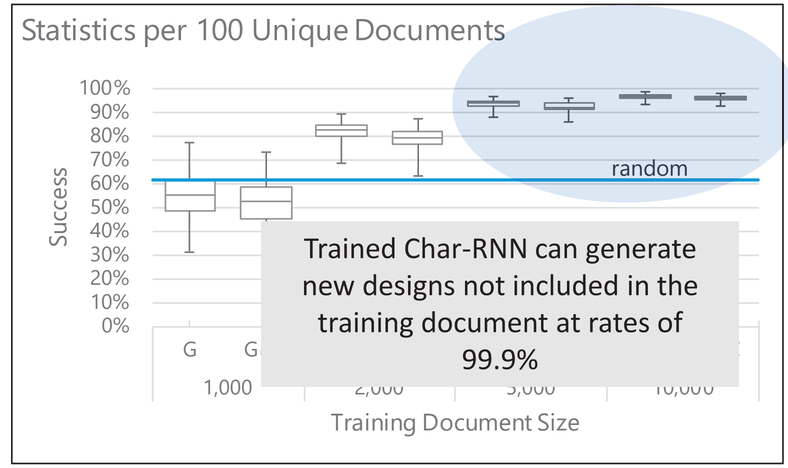
- Assembly layout
- Manifold geometry shape of each component (database ids for geometry files)
- Density of components
- Component scaling
- Control surface behavior

ex. h1_5_1 { s4_4_sy+ s2_2 s4 } { { f { r lc3_4_4 { f-- e f e e } } f { f hc2 { e e e f_sx e } } f } { e e { r lc4_4 { f e e { r hc4 { e f f e e } } e } } { r hc3 { e e f f f } } e } { e f e e f } }



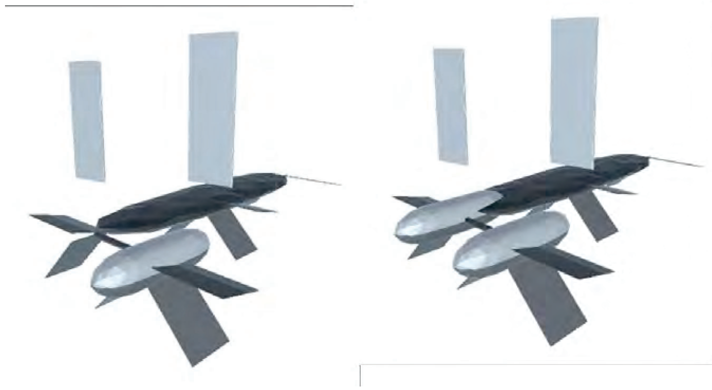
Teaching the AI the *language of feasible* design (syntax)

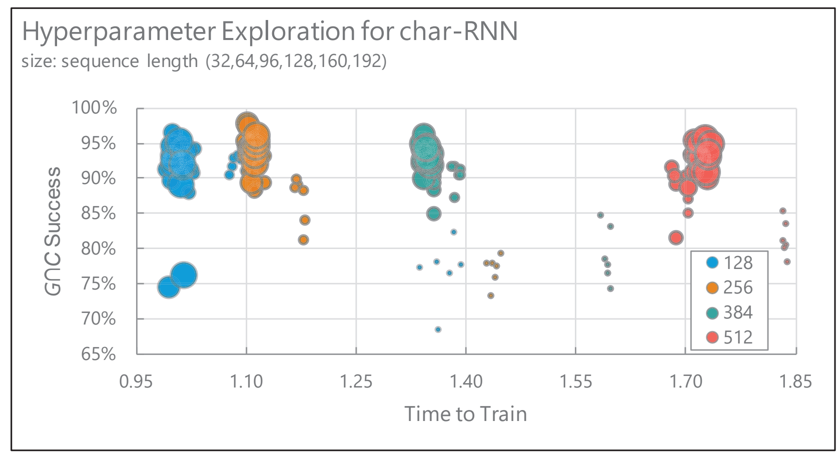
- Disallowance of collocation of components is a simple, but exemplar case
  - Codifying this is nontrivial (= HARD)
  - G ∩ C is *not* a context free grammar
- Designing the AI architecture is dependent on data and hyperparameter tuning



Feasible
```
h{sbn}{
  {ffee{rl{efeff}}}
  {efeff}
  {efeef}
}
```

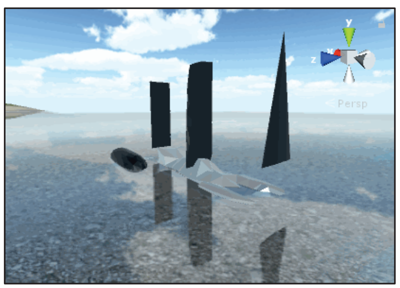
Infeasible (hull and front floater overlap)
```
h{sbn}{
  {ffee{rl{{rl{efeff}}feff}}}
  {efeff}
  {efeef}
}
```

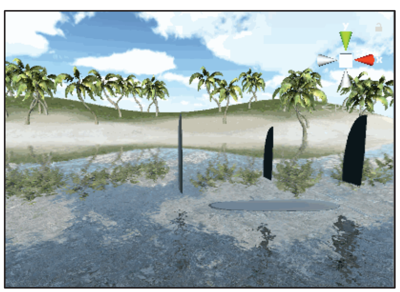


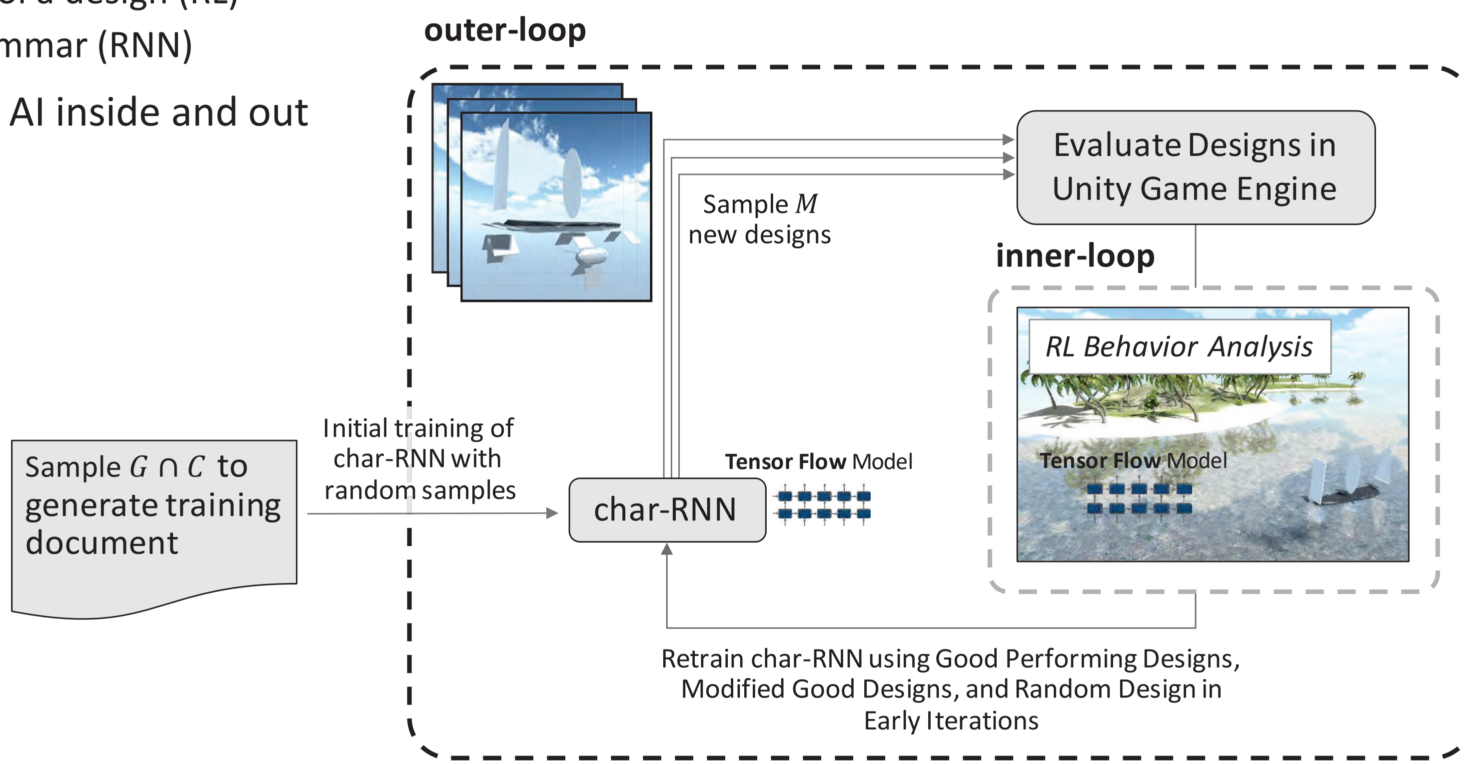


Framework to learn form and function

- Learn the feasible grammar (syntax)
- Learn the grammar of high-performing designs (semantics)
    - Inner-Loop learns to control a design (RL)
    - Outer-Loop learns the grammar (RNN)
- Optimize concurrently using AI inside and out







PennState
Applied Research Laboratory

Char-RNN Optimization Test Cases (no RL)

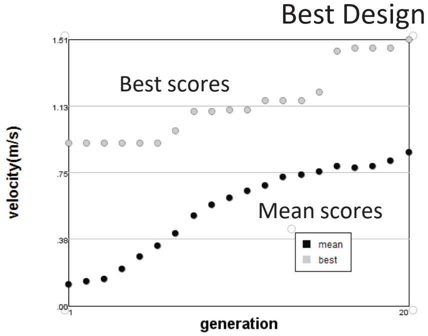
Goal : Show that the RNN optimizes the design by retraining itself on good performing designs from previous generations

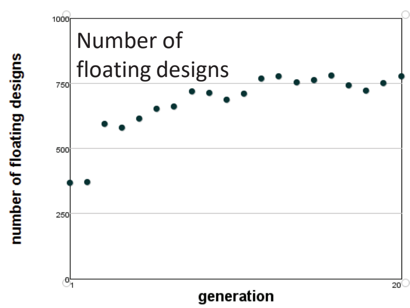
Problem and Optimization

Earth :
Crosswind :
No actuators,
No waves,
wind speed =
10 m/s

Goal : Maximize crosswind velocity

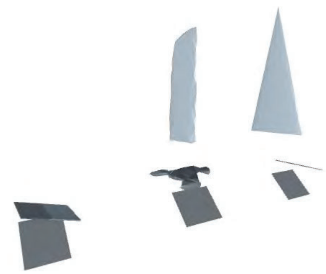
Best Design



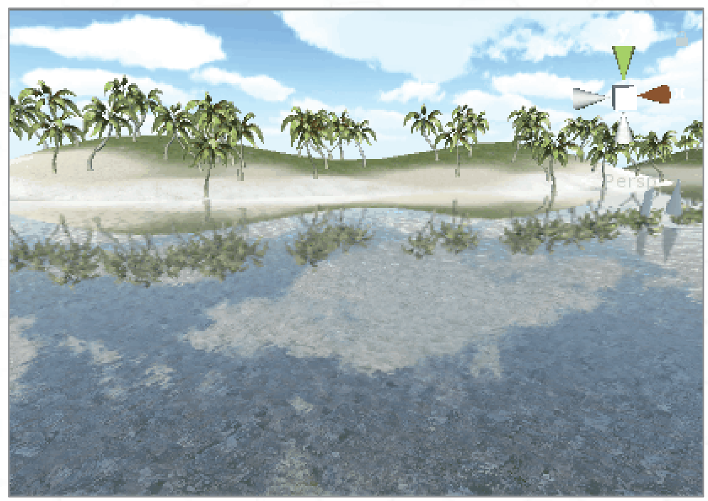


Learns the language of feasible design

Best Design = maximum velocity of 1.51 m/s

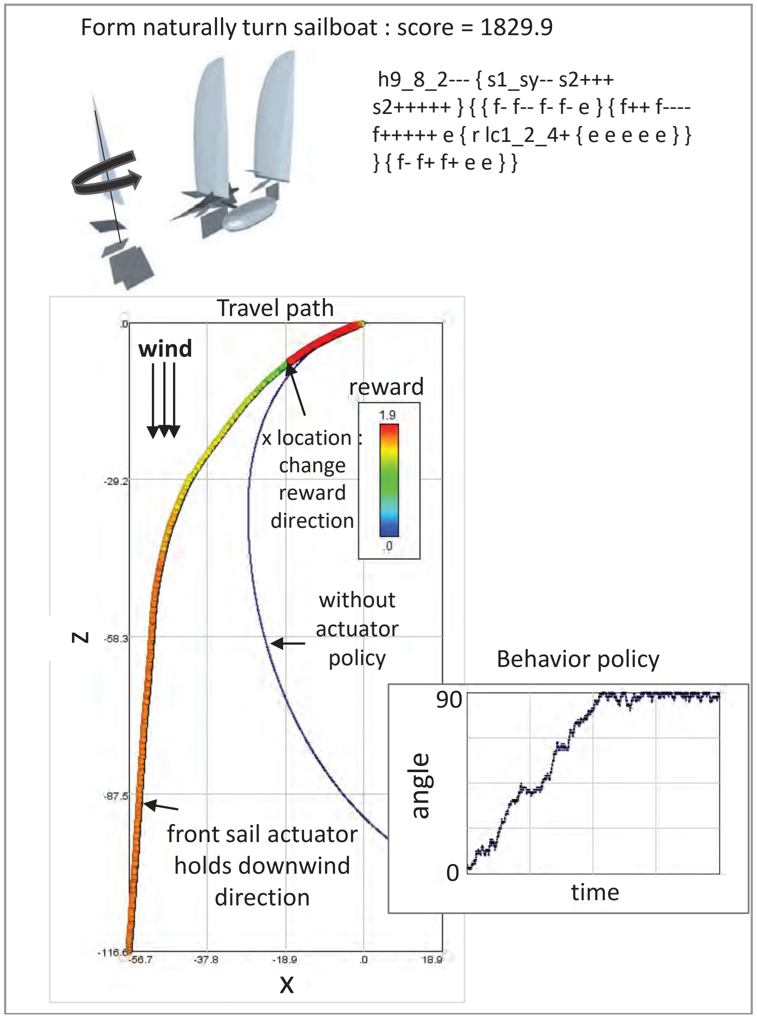


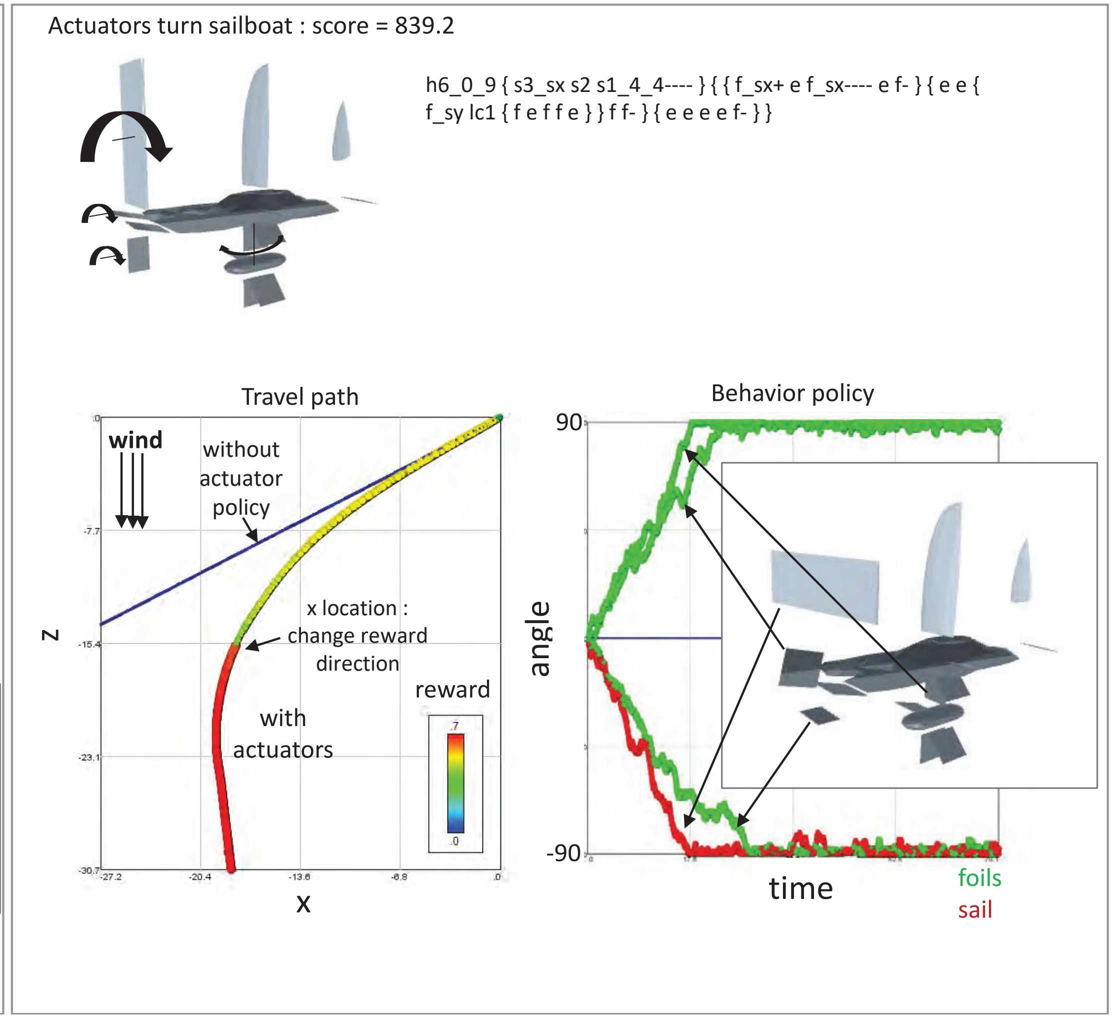
h8- { n s2_3_4--
s1+ } { { e e e f- f+
} { e e e f++ e } { e
e e f+ f+ } }



Char-RNN generates a sailboat that captures wind energy by lifting the boat using two back sails and minimizing the size of the hull to reduce drag

By learning both form and behavior, the AI develops a naturally turning boat that is augmented by the control policy

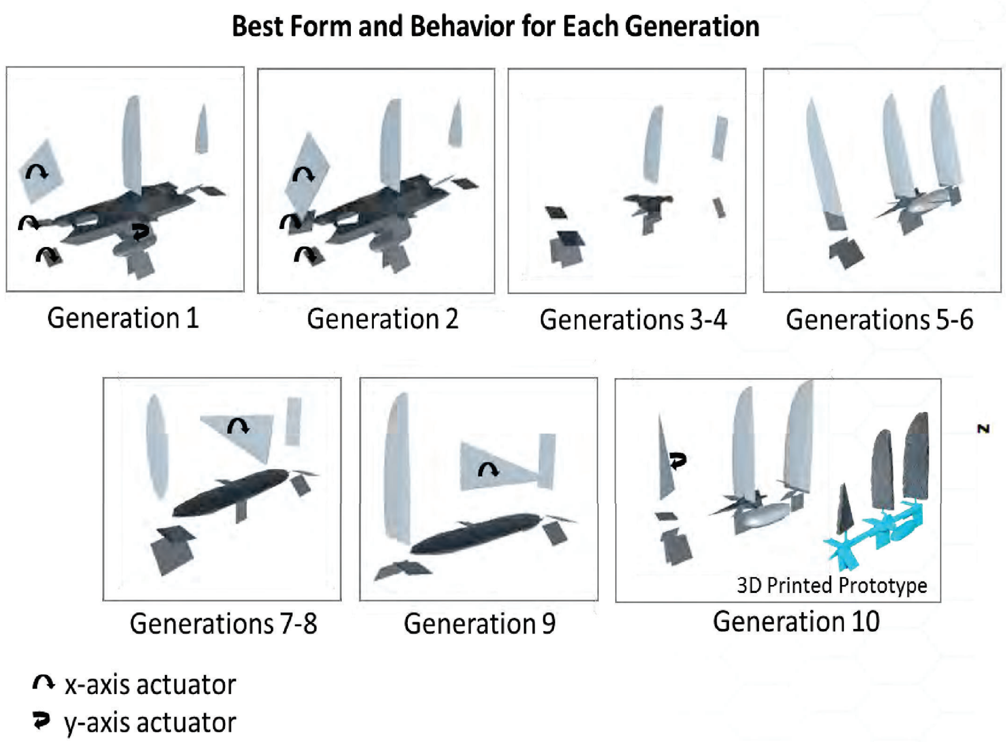


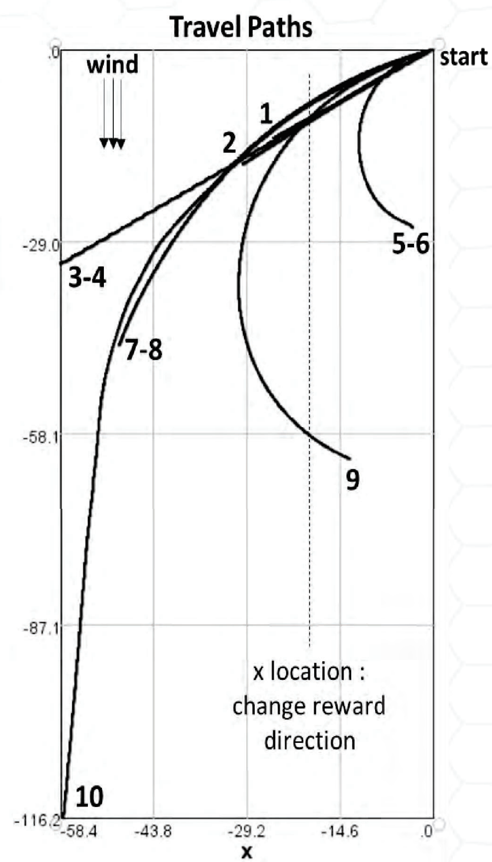


PennState
Applied Research Laboratory

Optimization with Actuated Surfaces using Reinforcement Learning

Shows the progression of best performing designs by generation

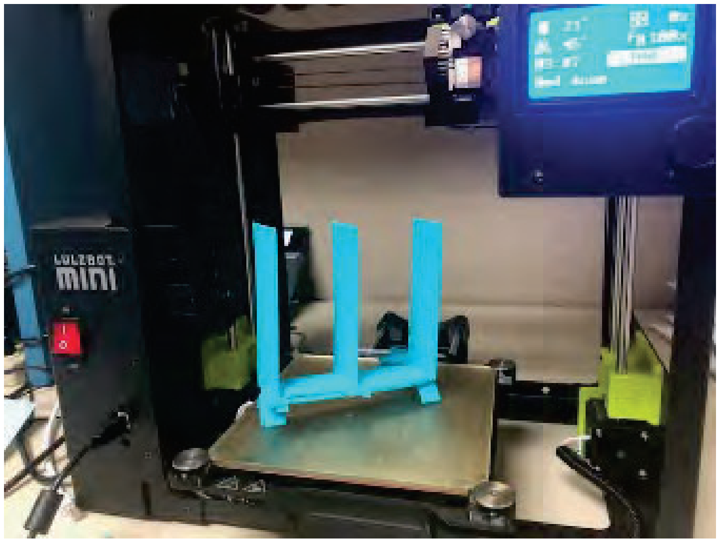


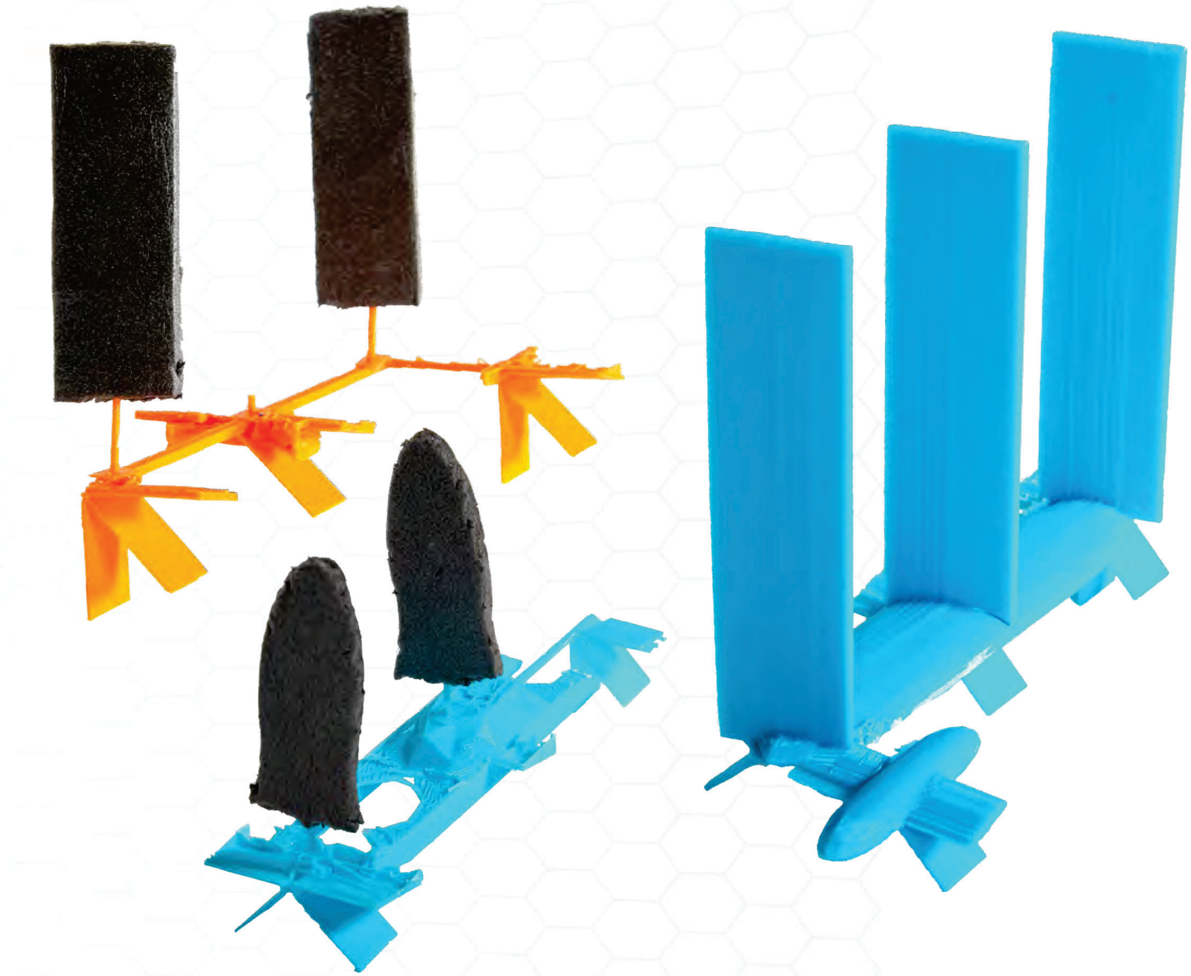


- Generations 1-4 : good crosswind behavior without a turn to travel downwind.
- Generations 5-6 : turn based on its form with no actuators.
- Generations 7-9 : turns with middle sail rotating horizontally about its x-axis.
- Generation 10 : uses one actuator sail, which rotates around its vertical axis, to straighten the turn

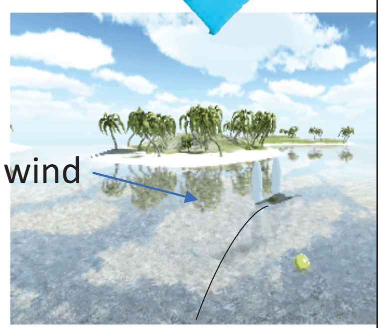
PennState
Applied Research Laboratory

3D Polymer Prints : Lulzbot Mini 2.0





Verify Designs Using Water Testing



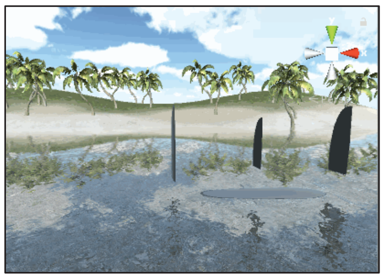
Simulated curve path

PennState
Applied Research Laboratory

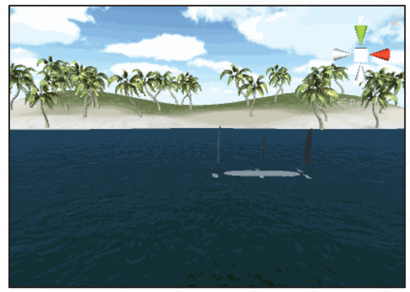
Verify Design Using Different Environmental Conditions

Test designs in multiple environments within a game engine for robust design

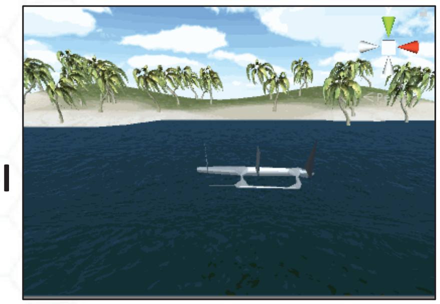
No Waves : Good Crosswind Performance

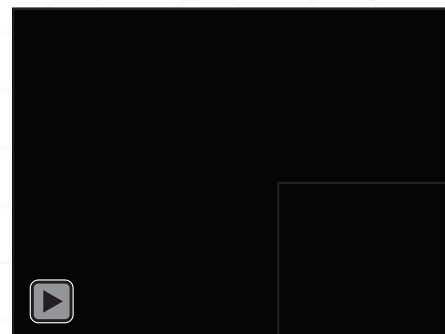


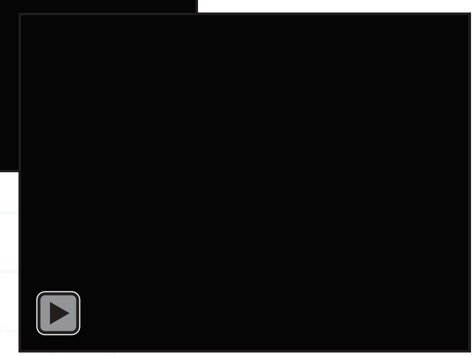
Waves : Capsizes



Char-RNN Optimizes Design for Crosswind Travel with Waves

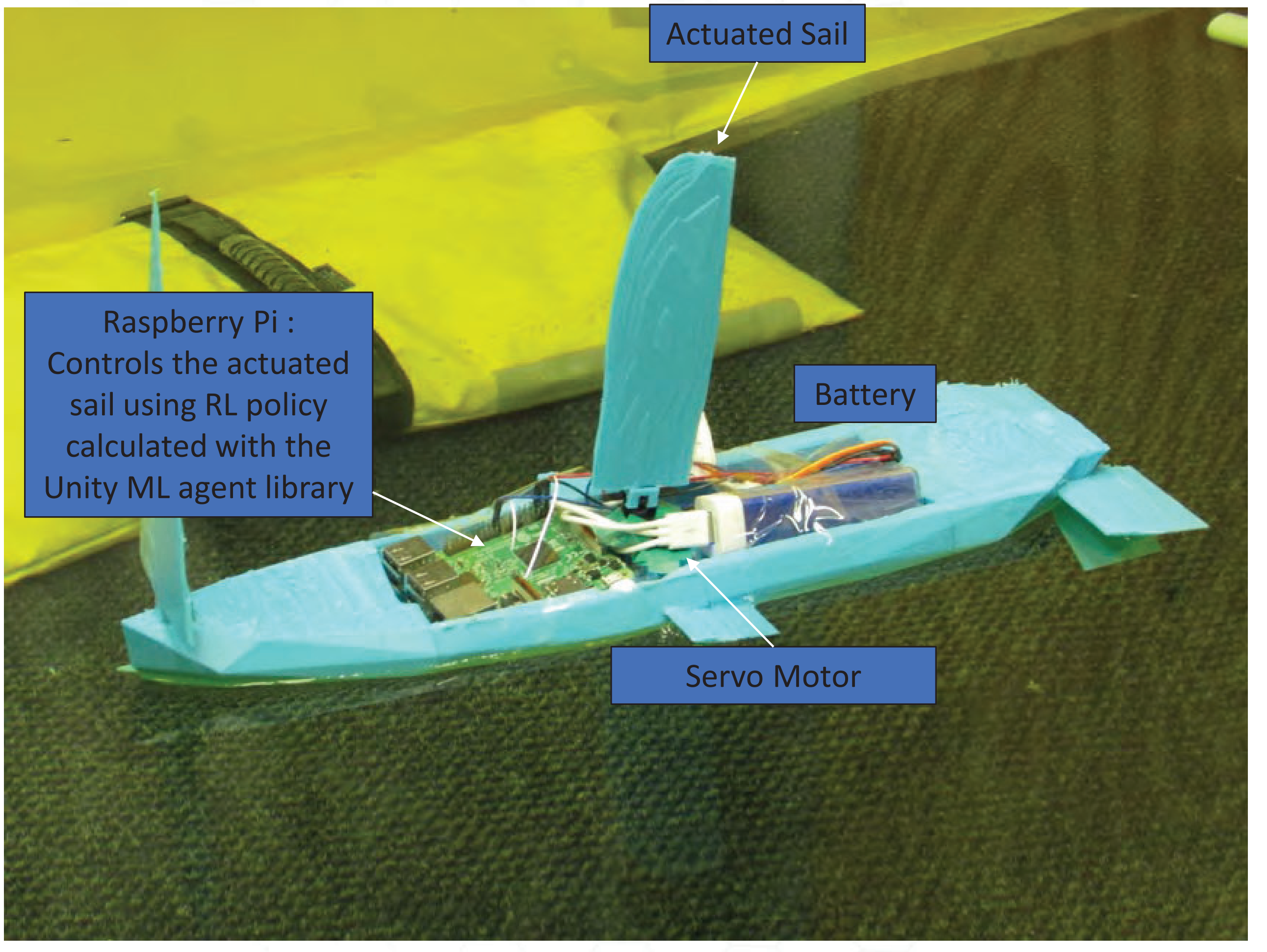






3D Printed Prototype with Actuated Sail

- Printed using polymer-based 3D printer (PLA)
  - Hull printed in 4 sections, based on build plate dimensional constraints
  - Sails are 3D printed as one piece
- Raspberry Pi 3 Model B
  - Python code controls angle positions for the servo motor
- Servo Motor
  - 180 degree turn capability
- Battery
  - Portable battery for the Raspberry Pi with a lifetime of 4 to 8 hours
- Waterproofed design by applying clear tape over hardware



3D Printed Prototype : Test in Pool

Used Python code on the Raspberry Pi to fix crosswind and downwind sail actuated positions, alternating 5 seconds at each orientation, based on control policy calculated in the Unity Game Engine

Behavior: Dynamic Soaring

Aircraft Example : Maximize Energy

Reinforcement Learning : Object learns behavior by observing a state (or environmental condition) and applies an action to maximize a reward

<u>Example</u>

Object : aircraft

State : velocity, height, orientation

Action : roll and/or pitch (min and max rates)

Reward : maximize kinetic and/or potential energy

PennState
Applied Research Laboratory

Aircraft Example : Travel Into the Wind

Changed the reward function to be a mix of excess energy and forward movement into the wind

AI generated a flight profile that was previously unknown (*SURPRISE!*)

Off-line analysis verified the feasibility of the discovered trajectory

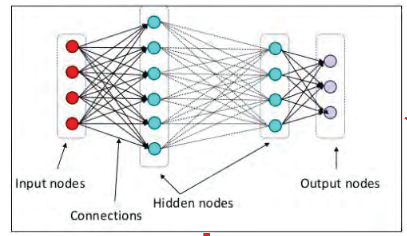
Demonstrates ability of AI to learn and generalize beyond known solutions

Behavior: Multi-rotors & transfer learning

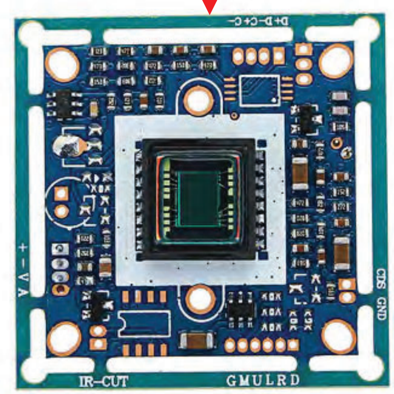
Impact: demonstrated trained Neural Network controller generalizing to multiple configurations (transfer learning)

Modify Design Based On Performance

Assemble Design

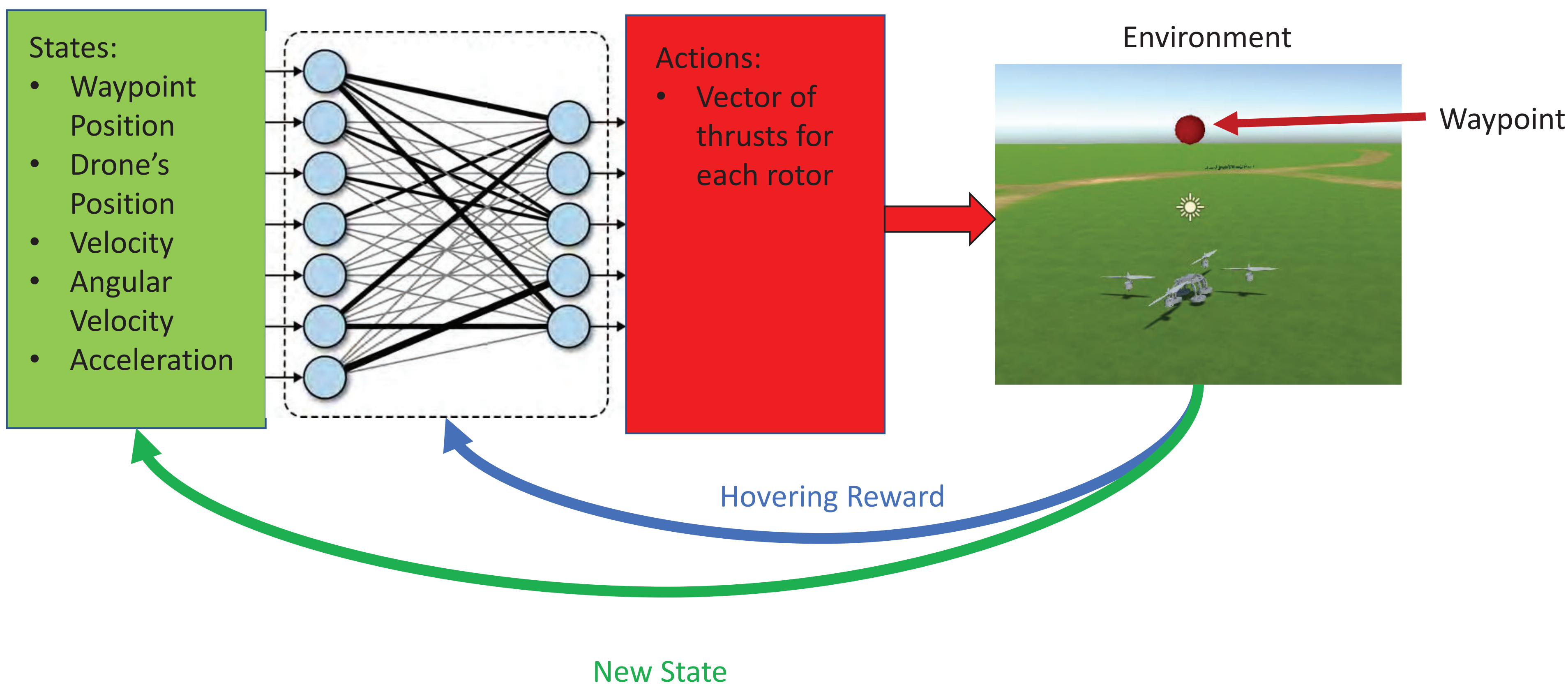


Transfer knowledge from this control algorithm to the next one



Tune/optimize control algorithm for configuration

Evidence of Claims: RL Training: Hovering with Quadcopter



Quadcopter Controller Easier Than Tricopter

Trained RL Quadcopter Controller From Scratch for 240k training steps

Trained RL Tricopter Controller From Scratch for 690k training steps

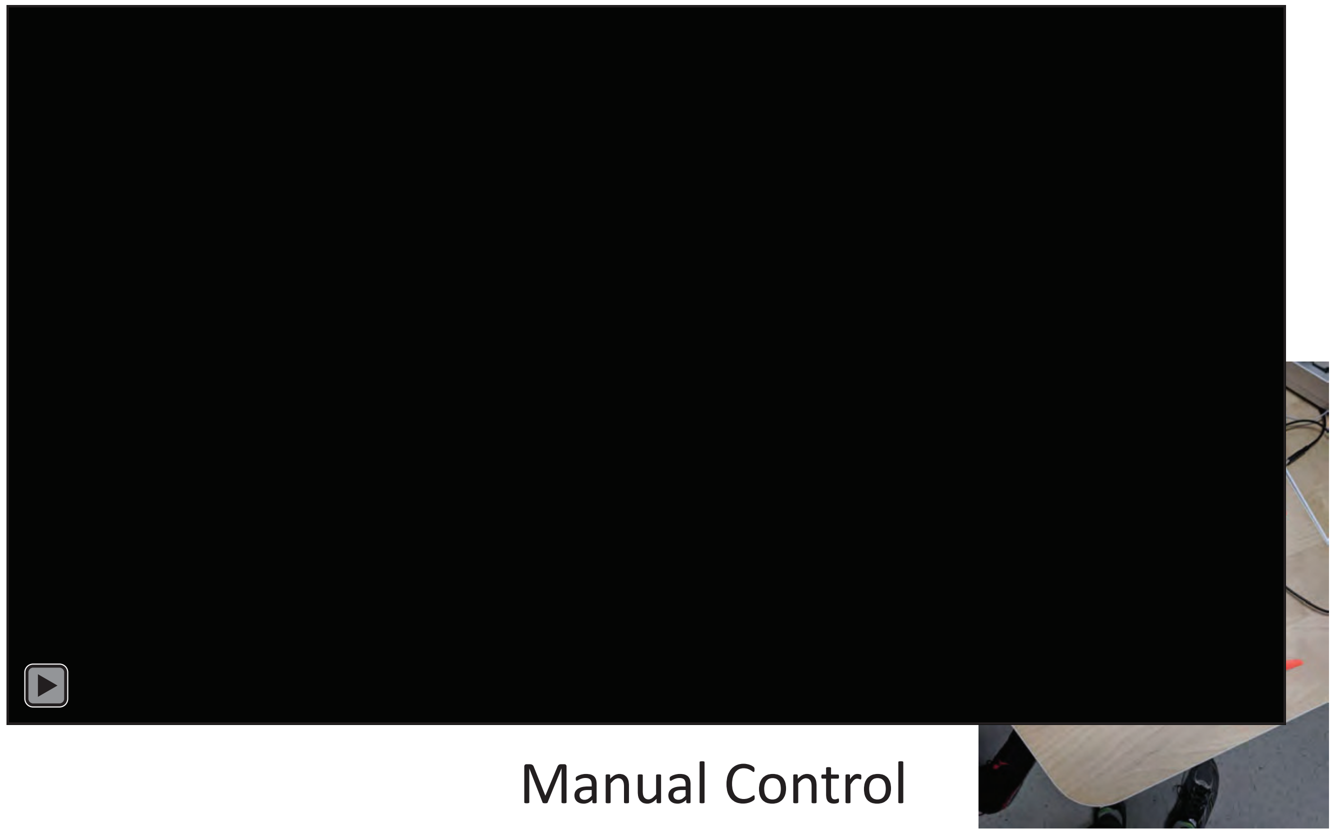
RL Transfer Learning Achieves Control of Tricopter

Initialized Tricopter Policy with a successful quadcopter policy (transfer learning), then trained for additional 340k steps

Ongoing Experiments: Transfer of trained AI Controller to real-world flight controller

Hypothesize that the AI ability to generalize controllers will lead to a robust control policy

Series of experiments to explore
- Robustness of trained controller
- Robustness with modeling error
- Robustness to failure



Manual Control

Summary: some key lessons learned *so far**

- **AI with video game engines can execute conceptual design rapidly, cheaply, and effectively**
- Crafting the reward function (the “carrot”) is key, and can and will lead to surprises
- Continued tension between max design freedom and moving off of “top dead center”
  - Total freedom: points clouds of geometry
  - Countably infinite: spatial grammars as tinkertoy systems
- When form and behavior are simultaneously optimized, you will see a sharing of reward achievement
- Training history matters: curriculum learning (easy then hard) can accelerate (or even make feasible) the training process
- AI demonstrates the ability to learn valid designs (syntax) that then perform well (semantics) in multi-physics problems for both point clouds and spatial grammars
- AI can interpolate within and *extrapolate beyond* the convex hull of the design space - *generate surprise*
- Critical part of the effort was in “reducing to practice” the concepts through physical prototyping,
  - Early and often, across domains
  - Accept a constrained design space that is implementable
  - Mesh the analyses with the testability
- Energy harvesting provides challenging problems: multi-physics, non-intuitive solution space

** NCE to 30 Sept 2019*

Things to Do

- Develop better understanding and methods to accommodate reward functions
  - Results of AI highly sensitive to reward functions – requires user expertise to achieve best results
  - Desire less human shaping of reward functions
  - moving more towards a biologically-inspired approach of rewards and penalties (e.g., digital version of human endorphins)
- Life-long learning: extend the curriculum training onto the physical platform
  - Off-line learning: post-mission assessment and update
  - On-line learning: failure mode response and recovery
- Using RNNs to address the *Spatial Grammar Maintenance Problem*
  - Feasible grammar more easily expressed as intersection of rules and constraints, which RNN can effectively learn
- Recommend choosing problems that are complicated enough to generalize from, but simple enough to build, and *reduce to practice*
  - The plan will not survive first contact with reality
  - Be prepared to be surprised throughout the process and respond
- Explore interaction between AIs, e.g., the RL for behavior and GAN for form
  - Currently interacting in the inner/outer loop process
  - Poorly understood
- Networks that can introspect on their own structure and their neighbor's
  - Network structure locks in potential
  - Could (for example) have the RNN or GAN tune the RL in the inner/outer loop optimization

Archival Products

Graduate Students

Matt Dering, PhD (2018): Generation and Evaluation of Designs using Deep Neural Networks

Conference Proceedings

Cunningham, J.D., Miller, S.W., Yukish, M.A., Simpson, T.W., & Tucker, C.S., (2019), Deep Reinforcement Learning For Transfer Of Control Policies, Proceedings of the ASME 2019 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2019), Anaheim, CA, IDETC2019-097689.

Wang, H. & Tucker, C.S., (2019), Pixel to Stroke Sketch Generation Using Reinforcement Learning, Proceedings of the ASME 2019 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2019), Anaheim, CA, IDETC2019-98481

Lopez, Christian E, Miller, Scarlett R, & Tucker, Conrad S, (2018), Human Validation of Computer vs Human Generated Design Sketches, Proceedings of the ASME 2018 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2018), Quebec City, Canada, DETC2018-85698.

Dering, M, Cunningham, J, Desai, R, Yukish, M.A., Simpson, T.W., Tucker, C.S., Stump, G.M., & Miller, S.W., (2018), A Physics-Based Virtual Environment for Enhancing the Quality of Deep Generative, Proceedings of the ASME 2018 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2018), Quebec City, Canada, DETC2018-86333.

Cunningham, J, & Tucker, C.S., (2018), A Validation Neural Network (VNN) Metamodel for Predicting the Performance of Deep Generative Designs, Proceedings of the ASME 2018 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2018), Quebec City, Canada, DETC2018-86299.

Journals

Stump, G.M., Miller, S.W., Yukish, M.A., Simpson, T.W., & Tucker, C.S., (2019) Spatial Grammar-based Recurrent Neural Network for Design Form and Behavior Optimization, Journal of Mechanical Design: Special Issue: Machine Learning for Engineering Design, MD-19-1169.

Cunningham, J.D., Simpson, T.W., & Tucker, C.S., (2019) An Investigation of Surrogate Models for Efficient Performance-Based Decoding of 3D Point Clouds, Journal of Mechanical Design: Special Issue: Machine Learning for Engineering Design, MD-19-1171.

In Review

Shu, D., Cunningham, J.D., Stump, G.M., Miller, S.W., Yukish, M.A., Simpson, T.W., & Tucker, C.S., (2019) A Physics-Based Virtual Environment for Enhancing the Quality of Deep Generative Designs, Journal of Mechanical Design: Special Issue: Machine Learning for Engineering Design.

Yukish, M.A., Stump, G.M., & Miller, S.W., (2019), Intersecting Generative and Constraint-Based Grammars via Recurrent Neural Networks for Design Creation, Journal of Mechanical Design, MD-18-1741.

Upcoming Submission

Cunningham, J., Simpson, T., and Tucker, C. (submission May 31st 2019). “Latent Space Exploration of Deep Generative Design Models”. Design Science Journal.

Fun Design ARL/PSU Team

- Head Nodes
  - Dr. Mike Yukish (ARL/PSU)
  - Dr. Conrad Tucker (PSU IE)
  - Dr. Timothy W. Simpson (PSU IE/ME)
- Hidden Layers
  - Mr. Gary Stump (ARL/PSU)
  - Dr. Simon Miller (ARL/PSU)
  - Mr. James Cunningham (PSU IE)
  - Mr. Matt Dering (PSU IE)
  - Mr. Dule Shu (PSU IE)
  - Dr. Jay Martin (ARL/PSU)
  - Charles Taylor III (ARL/PSU)
  - Ahkiel Moore (ARL/PSU , Calvary Scout National Guard)