AFRL-RI-RS-TR-2024-047

DESIGN.R - AI-ASSISTED CPS DESIGN: SYMBIOTIC DESIGN FOR CYBER-PHYSICAL SYSTEMS

VANDERBILT UNIVERSITY

MAY 2024

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2024-047 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
STEVEN DRAGER
Work Unit Manager

/ S /
NICK KOWALCHUK
RIT Chief Engineer
Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
MAY 2024	FINAL TECHNICAL REPORT	SEPTEMBER 2020	OCTOBER 2023

4. TITLE AND SUBTITLE

Design.R - AI-assisted CPS Design: Symbiotic Design for Cyber-Physical Systems

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
FA8750-20-C-0537	N/A	62303E

5d. PROJECT NUMBER	5e. TASK NUMBER	5f. WORK UNIT NUMBER
		R31S

6. AUTHOR(S)

Peter Volgyesi

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Prime: Vanderbilt University 1025 16th Avenue South, Suite 102, Nashville TN 37221

Sub: University of Alberta, 8625-112 Street Edmonton Alberta Canada T6G 2E1
Sub: University of Szeged,Bolyai Institute Aradi vertanuk tere 1 Szeged 6720 Hungary

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
DARPA 675 North Randolph St Arlington VA 22203-2114	Air Force Research Laboratory/RITA 525 Brooks Road Rome NY 13441-4505	AFRL/RI & DARPA	AFRL-RI-RS-TR-2024-047

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The goals of this Defense Advanced Research Projects Agency Symbiotic Design for Cyber-Physical Systems program were synthesizing, exploring, and optimizing multi-domain executable models with AI-based approaches.

To mitigate the lack of large and coherent engineering models - using the same design tools, language, and format - several tools and technologies were developed for generative synthesis and automatic evaluation of models and conceptual design considerations. Most importantly, we developed a constraint programming framework, which served as an integration platform for all these elements. By using a common design parameter/performance metric/constraint interface this approach enabled us to rapidly find feasible design alternatives and drive the exploration process towards the Pareto-front by using a pruning and pushing approach.

This overall architecture and the individual tools enabled us to rapidly adapt to changing domain constraints and mission requirements, and our team consistently delivered highly competitive results - compared to both human-engineered solutions and to other performers' designs - in both challenge domains.

15. SUBJECT TERMS

Cyber-Physical Systems, Design Space Construction, Design Composition, Design Space Exploration, synthesizing multi-domain executable models, optimizing multi-domain executable models

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
U	U	U	SAR	36

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
STEVEN DRAGER	N/A

PREVIOUS EDITION IS OBSOLETE.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 SUMMARY

Cyber-Physical Systems (CPS) are systems where the functionality emerges from the networked interaction of computational and physical processes. The tight interaction of physical and computational processes turns the design of these systems into a multi-domain co-design problem that integrates traditionally separated design domains into a coupled Design Space Construction (DSC), Design Composition (DC), and Design Space Exploration (DSE) process. Traditionally, these processes and the key design domains are targeted by separate groups of subject matter experts using highly different engineering approaches, design representations, and tools. These domain-focused models and fragile, semi-manual integration interfaces are one of the primary reasons for conservative choices in the overall system architecture and a very slow turnaround time in design iterations. Thus, design optimization is restricted to a small fraction of the feasible design space. The Defense Advanced Research Projects Agency's (DARPA's) Adaptive Vehicle Make (AVM) program was aimed at this problem by building horizontal models, tools, execution integration platforms, and end-to-end design automation tool suites. Our goals in this DARPA Symbiotic Design for Cyber Physical Systems (SDCPS) program were synthesizing, exploring, and optimizing such multi-domain executable models with artificial intelligence (AI) based approaches.

The most important challenge throughout the project was the lack of large and coherent engineering models - using the same design tools, language, and format. The CPS design space is highly fragmented - even compared to the software development domain. To mitigate these issues, we developed several tools and technologies for generative synthesis and automatic evaluation of such models and tried to capture conceptual design considerations (i.e., "back of the envelope" models - but still supported by exact tools and high-level parameters). The more refined models (e.g., computational fluid dynamics (CFD), finite element method (FEM)) required us to develop and train surrogate approximations to speed up the design exploration phase. Most importantly, we developed a constraint programming framework, which served as an integration platform for all these elements. By using a common design parameter / performance metric / constraint interface, the approach enabled us to rapidly find feasible design alternatives and drive the exploration process towards the Pareto-front using a pruning and pushing approach.

This overall architecture and the individual tools enabled us to rapidly adapt to changing domain constraints and mission requirements, and our team consistently delivered highly competitive results - compared to both human-engineered solutions and to other performers' designs - in both challenge domains.

The lack of readily available engineering models drove us to develop an alternative domain - beyond the core challenge problems of the program. Independent of the Unmanned Aerial Vehicle (UAV) / Urban Air Mobility (UAM) and Unmanned Underwater Vehicle (UUV) design applications, we developed a web-based electronic circuit design tool. By leveraging a large set of open-source circuit models, the tool was extended with AI-assistant capabilities to suggest new additions / missing elements throughout the design process. Even with an open-source dataset of ~5k samples, it is relatively modest to learn engineering patterns for this autocompletion task. Thus, this tool was extended with a human-assisted AI-based technology to rapidly parse circuit models from images or portable document format (PDF) datasheets for dramatically increasing the size of future datasets.

2.0 INTRODUCTION

Design Space Exploration and the synthesis of correct-by-construction designs require a large up-front engineering investment to produce a consistent library of multi-domain component models. One practical answer is to start this exploration with back-of-the-envelope calculations and incomplete high-level conceptual models. One of the prevailing approaches in practice is to capture the key design parameters, performance metrics, and fundamental constraints in spreadsheets. The value of such "design representation" should not be underestimated. It enables system-level human reasoning and very quick "what-if" analysis. Unfortunately, the spreadsheet approach breaks down fast as more and more design variables and constraints - rooted in physics and in the requirements - are added. Furthermore, design space exploration - by changing cell values and observing the ripple effects - is highly manual, error-prone, and follows a single linear exploration and evaluation path. Our constraint programming approach provides a more systematic approach by capturing the same conceptual level parameters and relationships of the model but automating with a massively parallelized model the exploration task. It also creates a bridge towards refining the conceptual design and integrating more detailed and accurate analysis models.

This framework allows for integrating multi-domain knowledge using symbolic expressions and / or using data-driven surrogate models. Thus, it is an ideal integration platform for driving the entire design space exploration process. As shown in Figure 1 and Figure 2, the domain, and mission-specific parts are "factored out" in the Graph Design phase, where domain-specific tools (SymCAD / SymDesign and UAVAnalyzer / GraphOps) translate the application domain and related surrogate models to the common language of the constraint programming framework.

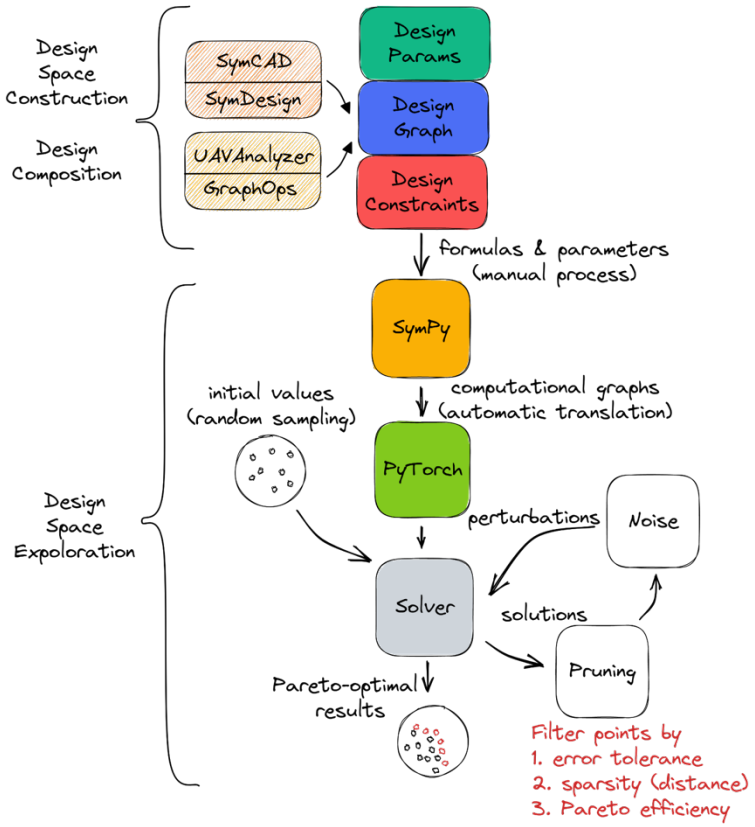
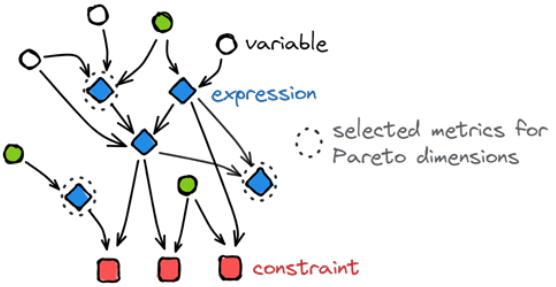


Figure 1. Design Space Exploration with Constraint Programming.

Our constraint solver framework is built on SymPy for capturing the formal expressions and constraints of the design. These expressions are automatically translated to PyTorch computational graphs for implementing a vectorized solver and design filters operating on thousands of design points concurrently. The solver and filter steps are executed iteratively (5-10 steps) and concurrently (1000-5000 design points) to find representative designs near the Pareto-optimal surfaces.



Definition (Design Space)

Bounded and continuous **design variables**: $x_i \in [a_i, b_i] \subset \mathbb{R}$. **Design space** $\bar{x} = (x_1, \ldots, x_n) \in S \subset \mathbb{R}^n$ given by system of non-linear equations

$$f_1(\bar{x}) = 0, \ldots, f_m(\bar{x}) = 0.$$

Inequalities $f(\bar{x}) \leq 0$ are modelled as equations: $\max(f(\bar{x}), 0) = 0$.

Algorithm (Simutaneous Mutivariate Newton-Raphson)

1. *Calculate the* **Jacobian** $J = (\partial f_j / \partial x_i)_{n \times m}$ *matrix numerically using backpropagation with* **Pytorch**.
2. *The Jacobian is* **almost always singular** *as most inequalities are already satisfied.*
3. *Calculate the* **Moore-Penrose inverse** J^+ *using SVD and zero out small singular values.*
4. *The* **approximation step** *is* $\bar{x}_{s+1} = \bar{x}_s - \bar{f}(\bar{x}_s) \cdot J^+$, *executed 10 times.*
5. *Execute the algorithm* **simultaneously** *for t independent designs.*

Definition (Evaluation Space)

A design space $S \subset \mathbb{R}^n$ is mapped to the **evaluation space** $T = \{\, \bar{g}(\bar{x}) \in \mathbb{R}^k : \bar{x} \in S \,\}$ using the $g_1(\bar{x}), \ldots, g_k(\bar{x})$ **evaluation metrics**. A design $\bar{x}_1$ **dominates** $\bar{x}_2$ if it is better in all metrics. A design $\bar{x}$ is on the **Pareto-front** of S if it is not dominated by any other design in S.

Algorithm (Genetic Algorithm for Pareto-Front Approximation)

1. *Generate a set of* **random designs**, *not necessary in the design space.*
2. *Use the* **Simultaneous Multivariate Newton-Raphson** *algorithm.*
3. *Keep those designs for which the previous algorithm has converged.*
4. *Keep those designs that are on the* **Pareto-front** *of the previous step.*
5. *Prune those designs that are too* **close together** *in the design space.*
6. *Add* **randomly mutated new designs** *so that we have the same number as in step 1, and go back to step 2.*

Figure 2. Multi-Objective Constrained Optimization.

We successfully applied the constraint-based approach in both SDCPS design challenges and used the iterative optimization process to find a large set of Pareto-efficient design points for various mission-level requirements. The benefits of the constraint programming approach are: (1) well-defined formal models even at the conceptual phase, (2) the incremental refinement of multi-domain constraints, (3) rapidly generating large sets of feasible designs with a vectorized solver, (4) iterative pruning of the design space for sparse, Pareto-optimal designs. Figure 3 shows a representative example of the Pareto-mapping process in the Southwest Research Institute (SwRI) Air Taxi (Hybrid or Electric) aeronautical Simulation (ATHENS) design problem.

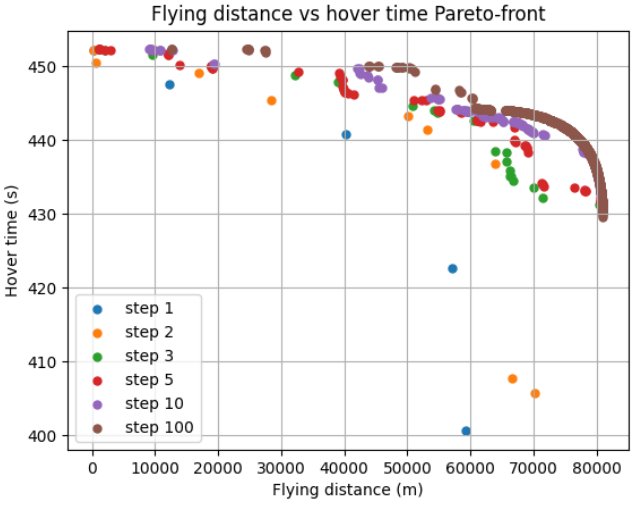


Figure 3. Example Pareto-front search results.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 SwRI / ATHENS CHALLENGE

Figure 4 shows the detailed workflow for UAV / UAM design ATHENS Challenge. The UAV analysis tool captures the most important domain concepts and properties of the available component library (batteries, motors, propellers, parametric wing and fuselage models) and by translating the high-level relationships among design parameters and system-level metrics (weight, lift, drag, thrust), it utilizes the constraint programming framework to identify valid and optimal combinations of these. The results are used by the GraphOps tool, which synthesizes the concrete vehicle assembly and executes the Southwest Research Institute analysis tools in an optimization loop. The current (refined) optimization uses a genetic algorithm-based approach due to the rugged and very highly sensitive error and performance surfaces.

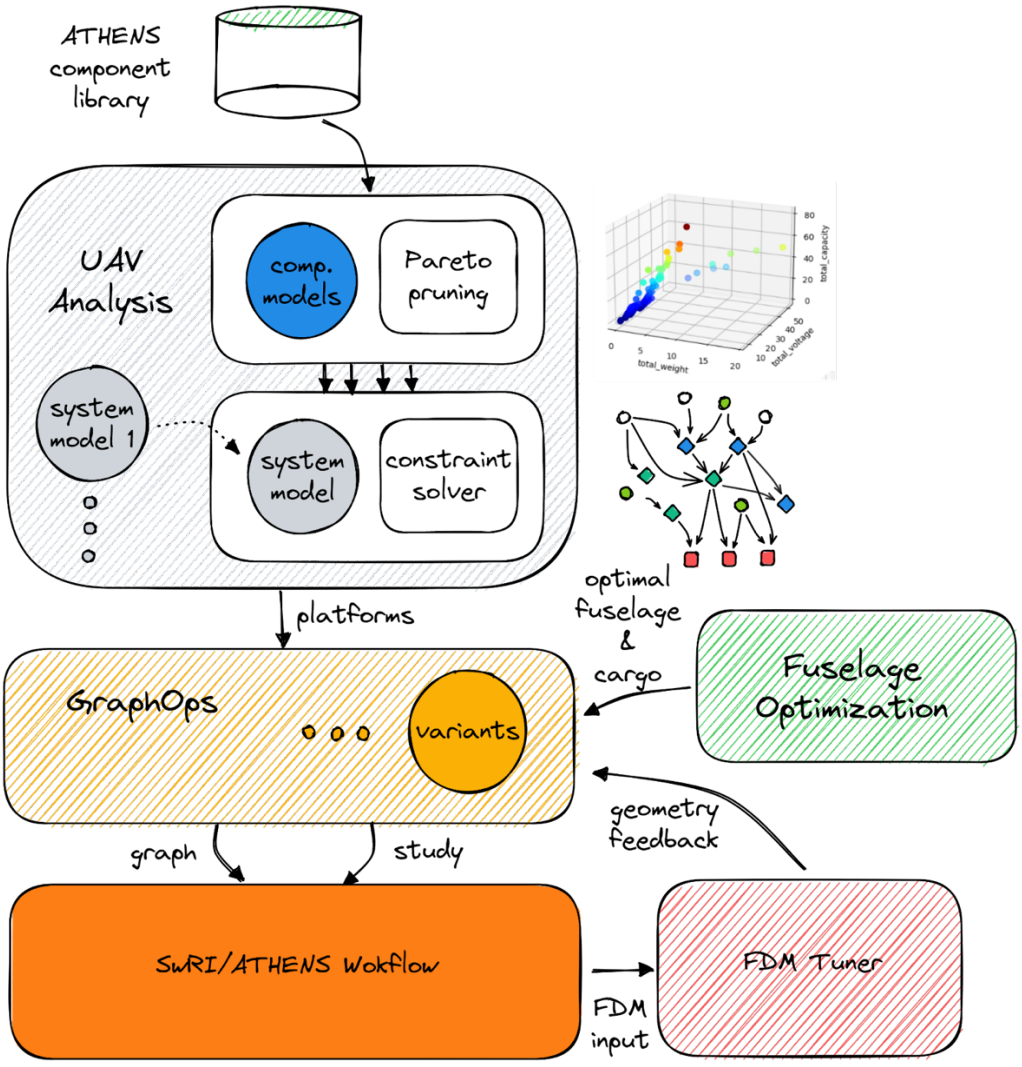


Figure 4. AI-assisted design workflow for UAV / UAM.

The UAV analysis tool, with example output shown in Figure 5, is a command-line interface (CLI) program, and its primary purpose is to find good candidates for downstream exploration. Thus, at this stage, we are not concerned about the concrete vehicle geometry, just highly approximate models, trying to find balanced solutions.

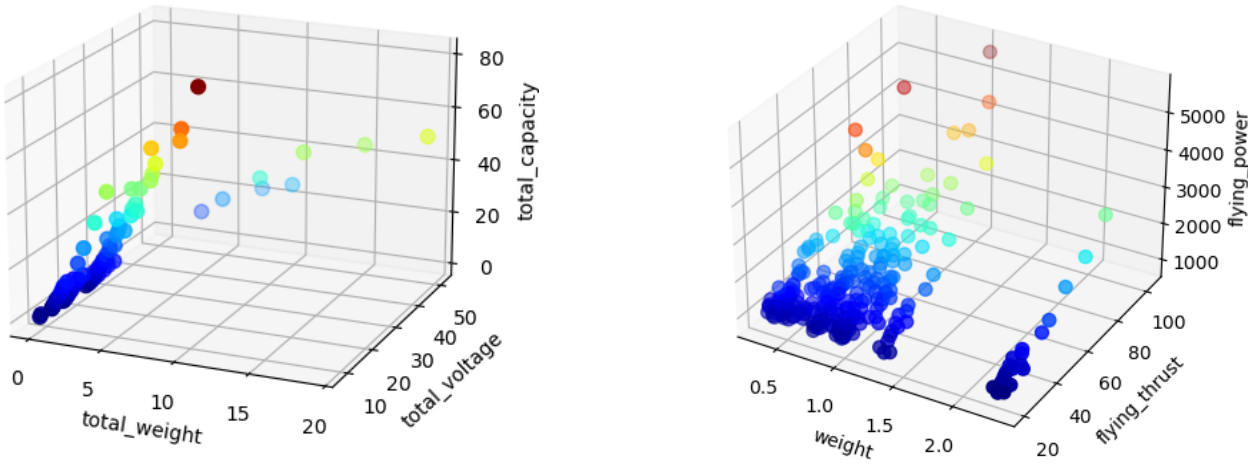


Figure 5. Pareto-front analysis for the SwRI UAV design corpus.

3.1.1 Design Space Construction / Design Composition

ATHENS-graphops is a tool designed to interface with the DARPA Symbiotic Design for Cyber-Physical Systems UAV corpus tools created by the Southwest Research Institute ATHENS team. The ATHENS tool retrieves specified UAV and air taxi designs from a JanusGraph setup and runs system performance simulations using integrated tools to measure the flight performance of the design and report the results of provided parametric studies.

The base corpus database (using JanusGraph) is provided by the ATHENS team and identifies the available components used to build the system. These components are assembled to create unique designs. The ATHENS-graphops tool is used to programmatically create these designs with user-developed platform specifications which indicate how the components are interconnected. Parametric values for design elements are identified to allow variation across multiple portions of the structure and to explore ranges of value adjustments within a system simulation run. Each platform definition can be designed to allow exploration of multiple variations of parts and their configuration values, with some limited topological variability (e.g., randomly choosing the number of propellors to attach and their positioning on the wings).

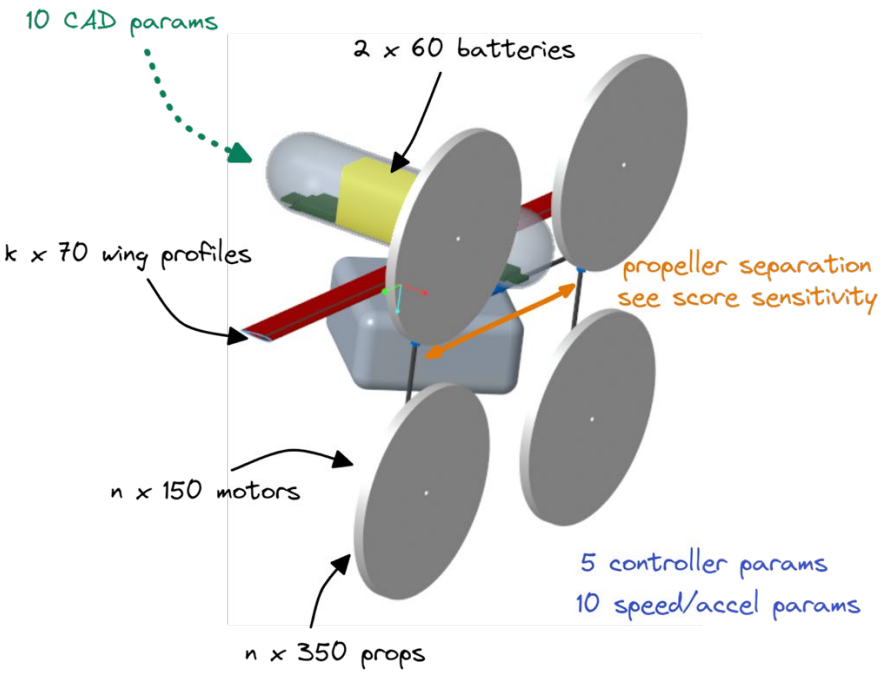


Figure 6. Combinatorial design space with the ATHENS corpus.

Given a platform definition, the base code will convert the desired design into a set of corpus components and build a design graph that can be retrieved with the ATHENS tools for evaluation. This base code set includes graph queries that manage core component connections and parameter configurations specific to each component type. Each component type is defined by a component schema with multiple options (sometimes hundreds) for known component types available (based on off-the-shelf components and their specifications), as shown in Figure 6. Using the schema, the ATHENS-provided corpus database can be scanned for inconsistencies to allow feedback to the SwRI team.

In addition to the designs that are created and placed in the JanusGraph database, parameter studies configuration files used by the ATHENS tool are comma separated value (.csv) files that can be created by defining a YAML Ain't Markup Language (YAML) file that outlines the desired flight parameters and the ranges of design parameters desired in the explorations. The tool also allows the specification of the workflow to run during evaluations of a platform design.

3.1.2 Design Space Exploration

The overall goal of this piece of the project, Stringer, was to design, implement, and utilize an efficient unmanned aircraft design system focused on using the generative power of a string-based grammar.

Specifically, this was broken down into three parts:

1. Creating a coherent and effective string-based grammar for the easy and concise description of unmanned aerial vehicles.

2. Creating a design actualization pipeline that accepts valid grammar strings and produces testable 3D-modeled designs, specifically for Creo Parametric (CREO).

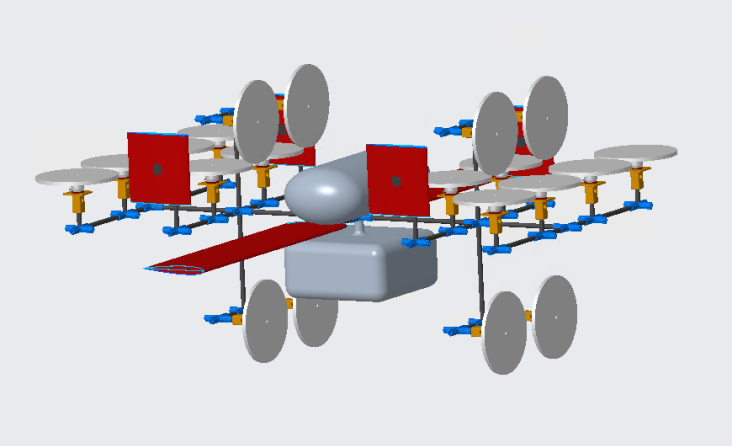
3. Creating a design space explorer to automatically generate thousands of unique designs without human input that are able to be processed by the aforementioned pipeline.

Methods:

- Goal 1 (string grammar): A detailed string grammar was designed, tested, and refined that was capable of expressing an incredibly large amount of vehicles of varying sizes and designs. The grammar can encapsulate the majority of the information regarding part type and orientation of a design in a string of 10-20 characters, depending on the size of the design. This grammar was utilized as the backbone for what was done in the rest of the project.

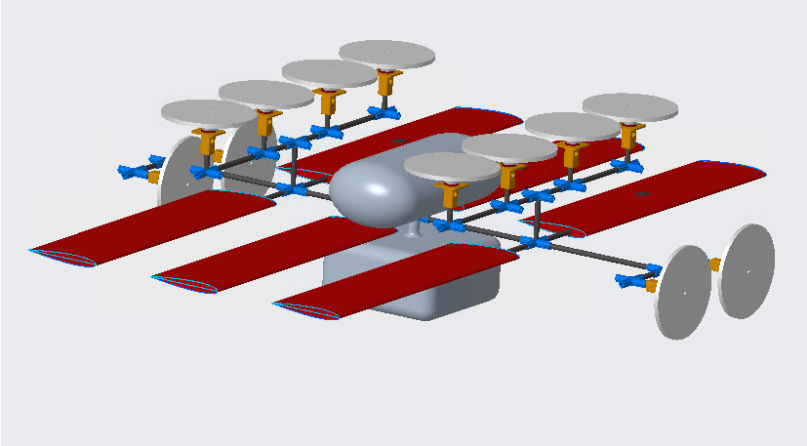
- Goal 2 (actualization pipeline): A fully automated pipeline consisting of an interpreter that would receive string definitions of designs and produce detailed JavaScript Object Notation (JSON) outputs was designed that was paired with an ATHENS-graphops designer that would accept these JSON files to create a testable three-dimensional (3D) model of the created design.

- Goal 3 (automated design exploration): A Programming in Logic (PROLOG) script was written for the automatic generation of valid strings as defined by the grammar, see Figure 7. Though this script was able to both generate valid strings and explore the space procedurally, methods for more efficient space exploration were being developed at the time of the project cancellation and could not be developed further.



Stringer grammar description:

[vvvv][lv(hh)(hh)vl][wfw][lv(hh)(hh)vl][vvvv]



Stringer grammar description:

[hh][w(vvvv)w][wfw][w(vvvv)w][hh]

Figure 7. Example vehicle architectures automatically generated by Stringer.

3.2 SYSTEMS & TECHNOLOGY RESEARCH LLC (STR) / NAUTILUS CHALLENGE PROBLEM

Using the constraint programming framework we successfully integrated several key engineering domains (mechanical, structural, CFD, propulsion / energy, control / navigation) to deliver and adapt optimal solutions to rapidly changing mission requirements.

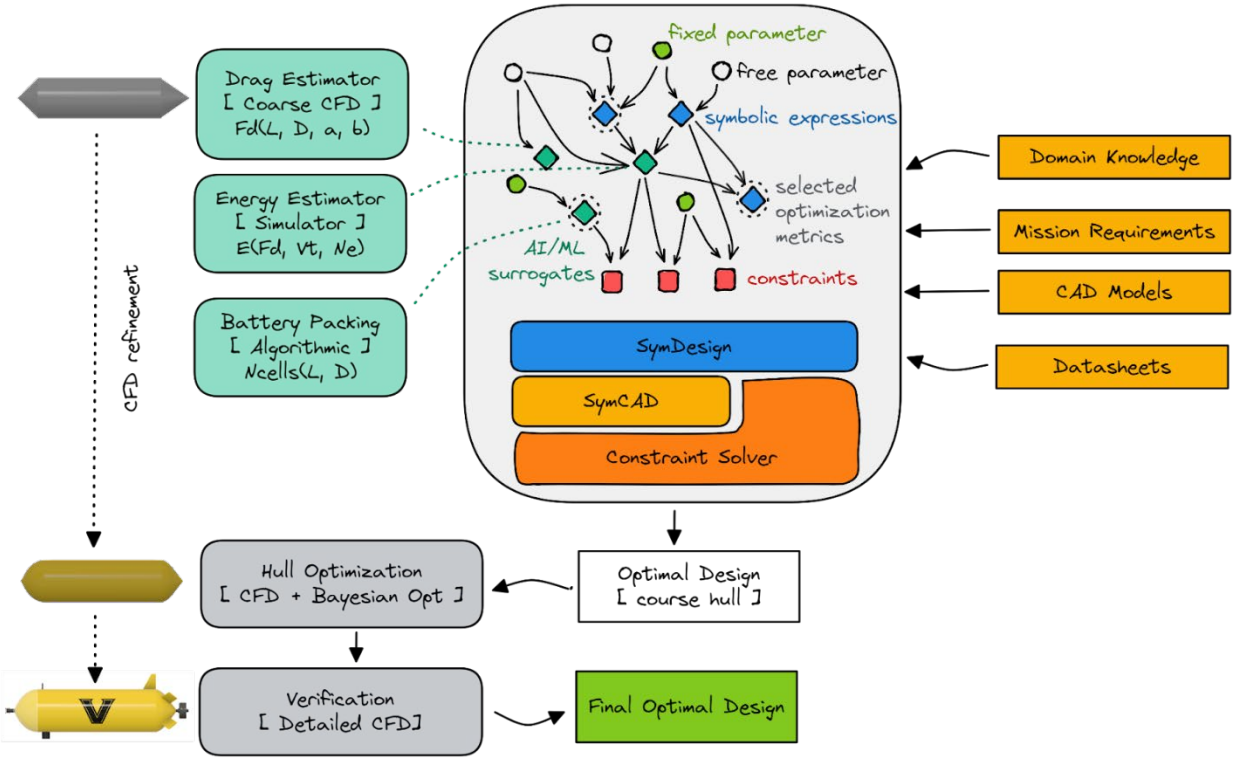


Figure 8. AI-assisted Design Workflow for UUV.

As shown in Figure 8, the developed core technology tools were SymCAD - a generic constrained geometric assembler, SymDesign - vehicle assembly driven by domain and mission constraints, a multi-fidelity level CFD surrogate and optimization toolbox, a structural strength surrogate model based on FEM and using physics-guided neural networks (NN). Additional domains (optimal battery packing, and total energy estimation based on Monte-Carlo simulation runs) were handled with simpler model fitting approaches.

This automated toolchain was capable of evaluating hundreds of design candidates in a matter of minutes and generating the final Computer Aided Design (CAD) assembly model of the vehicle - using a fixed set and concrete and parametric component models.

3.2.1 Design Space Construction / Design Composition

3.2.1.1 SymCAD

SymCAD is a Python library that combines symbolic model creation, orientation, and assembly with concrete CAD representations and manipulations. It allows users to programmatically design individual shape-based parts, ranging from the very simple and generic to the complex and specific, while allowing parameters related to the geometry, orientation, and placement of a part to be expressed symbolically.

The library allows for use of either scripted or modeled parts, where a *scripted SymCAD part* is a part whose CAD representation is generated programmatically using the FreeCAD Python backend. A *modeled SymCAD part* is a part whose CAD representation is pre-created by the user

in the FreeCAD format and stored in a well-known directory. Any existing CAD model can also be imported and used as a SymPart within SymCAD as if it were a built-in part. If the existing CAD model is in the FreeCAD format and follows a specific set of rules, the model can even be parametric and contain symbolic free parameters.

The following list of physical properties may be retrieved from a SymCAD part at any time by simply requesting the property method on the corresponding part: mass, material volume, displaced volume, surface area, unoriented center of gravity, oriented center of gravity, unoriented center of buoyancy, oriented center of buoyancy, unoriented length, oriented length, unoriented width, oriented width, unoriented height, and oriented height.

Two assembly modes are supported for the creation of composite parts or assemblies: assembly-by-placement and assembly-by-attachment. Assembly-by-Placement is used to allow for complete control over the precise placement and orientation of every part within an assembly. This mode is the default for all SymPart components which do not contain predefined attachment points or explicit connections to other parts. In this case, the placement of a part will be represented by three symbolic parameters representing the origin / center-of-placement of the part in its own coordinate space, as well as three additional symbolic coordinates representing the placement of the origin of the part in the global coordinate space.

Assembly-by-Attachment is used to automate the process of placing rigidly attached parts within an assembly. Using this placement method, the intricacies of part placement can be relegated to the SymCAD library, which may greatly simplify the number of free variables in the resulting symbolic assembly. In order to use this methodology, individual SymPart components must define one or more *attachment points*, which defines a point in the local coordinate system of the part that can be rigidly attached to other parts. The available properties for an assembly include mass, material volume, displaced volume, surface area, center of gravity, center of buoyancy, length, width, and height.

Once created, it may be necessary to obtain the physical or geometric properties of only a subset of the parts within an assembly. In order to achieve this, parts can be grouped into so-called *collections*. Once added to one or more collections, the geometric properties of the SymPart components contained in any number of these collections may be accessed by specifying the collections of interest in the corresponding property accessor of the assembly object.

The approach that SymCAD takes to achieve its above-stated goals includes:

- Utilizes the Python sympy library for symbolic manipulation of parameters.
- Utilizes the FreeCAD Python backend for CAD file processing and manipulation.
- Works with both scripted and modeled CAD designs.
- Provides two methods of model construction:
    1. Assembly-by-Placement: Part placement is explicitly defined.
    2. Assembly-by-Attachment: Part placement is implicit based on rigidly attached parts.
- Includes a built-in library of generic parts.
- JSON-based Graph application programming interface (API) for design representation.
- Specification of center-of-placement / origin for each part.
- Custom attachment and connection points for each part.
- Part-based physical property retrieval:
    1. Based on closed-form equations (concrete or symbolic).
    2. Based on CAD representations (concrete).
    3. Based on pre-trained neural networks (concrete or symbolic).
- Assembly-based cumulative physical property retrieval.
- Physical properties include: mass, material volume, displaced volume, surface area, center of gravity, center of buoyancy, length, width, and height.
- Part importation from existing CAD models (FreeCAD, Standard for Exchange of Product Data (STEP), or Stereolithography (STL)).
- Interference detection for parts within an assembly.
- Easy-to-create custom parts (scripted or modeled).
- Automatic separation of regular and displacement models.
- Parts and assemblies exportable to FreeCAD, STEP, or STL.
- State-based physical properties for assemblies.
- Simple interface for concretizing free parameters in a symbolic design.
- Symbolic parameters can auto-combine or concretize based on attachments.
- Auto-generated and updated documentation upon GitHub commit (documentation located at https://symbench.github.io/SymCAD).

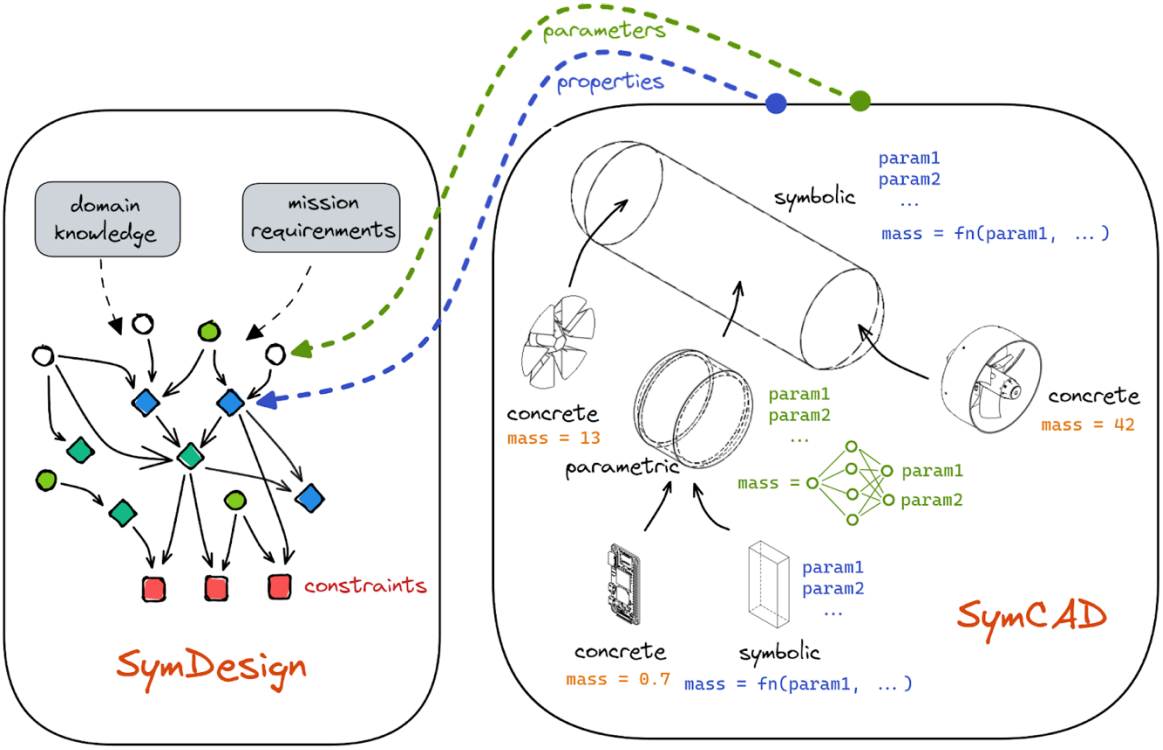


Figure 9. Design Space Construction w/ SymDesign & SymCAD.

3.2.1.2 SymDesign

The process of Design Space Construction is shown in Figure 9, where SymDesign is a Python-based tool for exploring a feasible design space given a set of mission-based requirements, an available listing of parts, and any user-specified design constraints. It makes use of both SymCAD and the Constraint Solver Tool to quickly generate a large number of concrete designs (vehicle-based in the current state) along with their corresponding CAD models and mass properties (both static and dynamic).

The expected process for using this tool is the following:

- Define a set of mission parameters and requirements based on the distinct "mission segments" that a target vehicle is expected to navigate.
- Enumerate the list of possible concrete sensors and attachments that are available for use on the target vehicle.
- Specify a set of possible discrete parameter options for various components (e.g., battery cell type and size, engine type, etc.).
- Auto-instantiate a set of feasible symbolic representations for the target vehicle (i.e., generic ordering and placement of components may be indicated, but concrete placement values, attachments, sizing, and other properties remain symbolic).
- Generate design point clouds that satisfy the mission requirements, which allows you to:
    - Enable, disable, or alter free variables.

- Change search bounds.
- Add or remove constraints.
- Explore trade spaces and design spaces.
- Regenerate point clouds at increased levels of resolution.

The above process is made possible through the use of "hierarchical symbolic design" in which every component in the design process is derived from a general base "Component" class with increasing levels of concreteness. For example, second-level derived component classes might include "attachment", "buoyancy component", "pitch component", "roll component", or "static component", where each class defines additional characteristics and derived values that all subcomponents will share (for example, the displaced volume of a buoyancy component at its minimum, neutral, and maximum buoyancy levels).

The shape, size, and assembly of all components is relegated entirely to the SymCAD tool, and the constraint-based solution of constraining equations and requirements is relegated to the Constraint Solver Tool. In essence, this tool provides the glue between defining mission requirements, obtaining symbolic equations for feasible designs and models, and solving a system of equations that satisfy all specified constraints and requirements.

The approach taken by this tool is to provide a clear separation of concerns regarding: 1) design representation, 2) constraint solution, and 3) surrogate modeling. All design representations and modeling are handled by the SymCAD library, all constraint solving is handled by the Constraint Solver Tool, and all surrogate modeling is handled within this tool itself.

The surrogate models currently implemented in this tool include: oceanic and atmospheric models, packing models, pressure vessel models, and lift and drag (CFD-based) models.

The oceanic models are National Oceanic and Atmospheric Administration (NOAA) based models allowing for the calculation of gravitational acceleration based on latitude and pressure at arbitrary ocean depths, water density at arbitrary pressure, temperature, and salinities, freshwater absolute viscosity, and seawater absolute and kinematic viscosity. The packing model is currently only used to pack cylindrical shapes (like batteries) and is based on a trained neural network dedicated to this task. The pressure vessel models use well-known equations (such as hoop-stress equations, etc.) for determining the minimum required cylindrical thickness, spherical thickness, or flat endcap thickness based on material type, maximum depth, and pressure vessel radius. The lift and drag models are currently based on neural networks trained on the output of computationally expensive CFD models for a known set of fairing shapes. Each of these models has a well-defined interface, allowing for them to easily be replaced, updated, or changed without affecting the runtime properties of the tool.

Using these surrogate models along with symbolic design representations generated by SymCAD, we can create a set of symbolic constraint equations which are passed to a Newton-Raphson-based solver within the Constraint Solver Tool to both solve for and optimize a large number of valid designs, optionally identifying and pruning the design space based on one or more Pareto fronts, as specifiable by the user.

3.2.2 Design Space Exploration

3.2.2.1 Constrained Bayesian Optimization for UUV hull design

The process of optimizing the hull design for Unmanned Underwater Vehicles, as shown for our case in Figure 10, is a multifaceted engineering task aimed at producing a UUV hull with optimal characteristics tailored to specific requirements. This procedure first necessitates the incorporation of complex engineering simulation tools that are computationally intensive. Secondly, it demands the amalgamation of an optimization framework that efficiently utilizes sample data with the comprehensive suite of integrated tools. In pursuit of this, we have amalgamated the computer-aided design software, FreeCAD, with the computational fluid dynamics tool, OpenFOAM, for the automated assessment of designs. For the optimization phase, Bayesian optimization (BO) was selected. BO is recognized for its effectiveness in optimizing simulations that are both costly and time-intensive, and it is notably efficient in its sample use, as evidenced in various scenarios such as tuning of hyper-parameters and design of experiments. The optimization sequence is crafted to accommodate constraints for unviable designs within the process. By merging a domain-specific suite of tools with AI-driven optimization, we have automated the hull design optimization for underwater vehicles. To empirically assess our approach, we applied it to two distinct real-world underwater vehicle design cases to confirm the functionality of our tools. The scripts for conducting the experiments and the necessary steps for setting up the toolchain are available at this GitHub repository: https://github.com/vardhah/ConstraintBOUUVHullDesign.

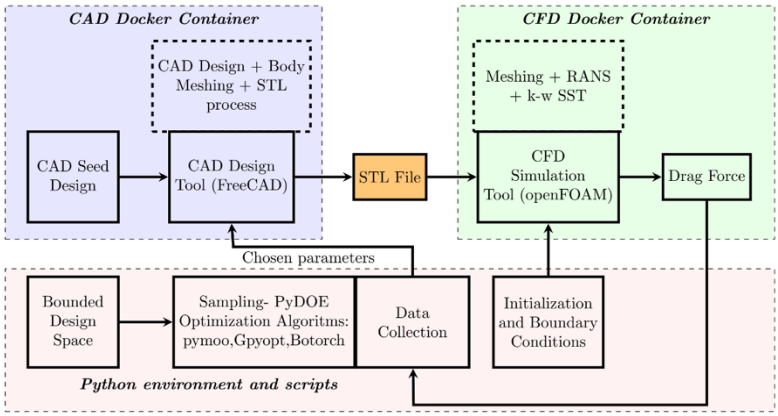


Figure 10. Constrained Bayesian Optimization for UUV hull design.

3.2.2.2 Sample efficient surrogate modeling

In computer-aided engineering design optimization, particularly when dealing with simulators that are known for their complexity and lengthy computation times, a common strategy is to employ a surrogate model. This model serves as a less computationally expensive approximation of the simulator's performance. The primary hurdle in developing such surrogates is the extensive data required, which necessitates running resource-intensive simulations.

To address this, Active Learning (AL) strategies are employed, which are designed to understand the relationship between inputs and outputs while reducing the need for labeled samples. The prevailing method in AL, especially for regression tasks, is Bayesian learning, which can be computationally demanding, especially when paired with Deep Neural Networks (DNNs). Despite this, DNNs are adept at capturing highly non-linear relationships in complex and high-dimensional data sets.

In an effort to harness the superior learning abilities of DNNs, while avoiding the computational intensity of Bayesian methods, we introduce a straightforward and scalable active learning technique. This method cleverly determines which samples require labeling through strategic selection. It operates on a student-teacher framework to educate the surrogate model. Using our novel method, we have managed to attain comparable levels of accuracy in our surrogate models with up to a 40% reduction in the number of required samples compared to traditional methods. Our method has been empirically tested across various engineering applications of interest, including finite element analysis, computational fluid dynamics, and propeller design.

3.2.2.3 Anvil

Our contribution is the development of an open-source integrated CAD-CFD solution by combining FreeCAD for CAD modeling and OpenFOAM for CFD analysis. This solution is further augmented with AI-powered sample-efficient optimization methods such as Bayesian optimization among other algorithms. We have named this toolkit Anvil, positioning it as a versatile scientific machine learning instrument for shape optimization, operable in data generation, CFD evaluation, and shape optimization modes. In the data generation mode, Anvil autonomously executes CFD studies and collates data for surrogate model training based on a parametric CAD seed design, set design space boundaries, computational limits, and predefined budgets. In the optimization mode, it navigates the design space to pinpoint the optimal shape, guided by specified objectives, constraints, and budget allocations. To utilize Anvil, users must supply a JSON configuration file along with a parametric CAD seed design. Anvil is designed to examine solid-fluid dynamics under any subsonic flow conditions, whether underwater, terrestrial, or aerial. A notable advantage of Anvil is its Python integration, which facilitates the incorporation of additional optimization and sampling strategies like GFlowNet, Open Multi-Disciplinary Design Analysis and Optimization (OpenMDAO), and others. We validate the utility of Anvil across various simulation scenarios and for an optimization case study. The open-source code, installation instructions, resources such as CAD seed designs and sample STL models, results from experiments, and comprehensive documentation can all be accessed at https://github.com/symbench/Anvil.

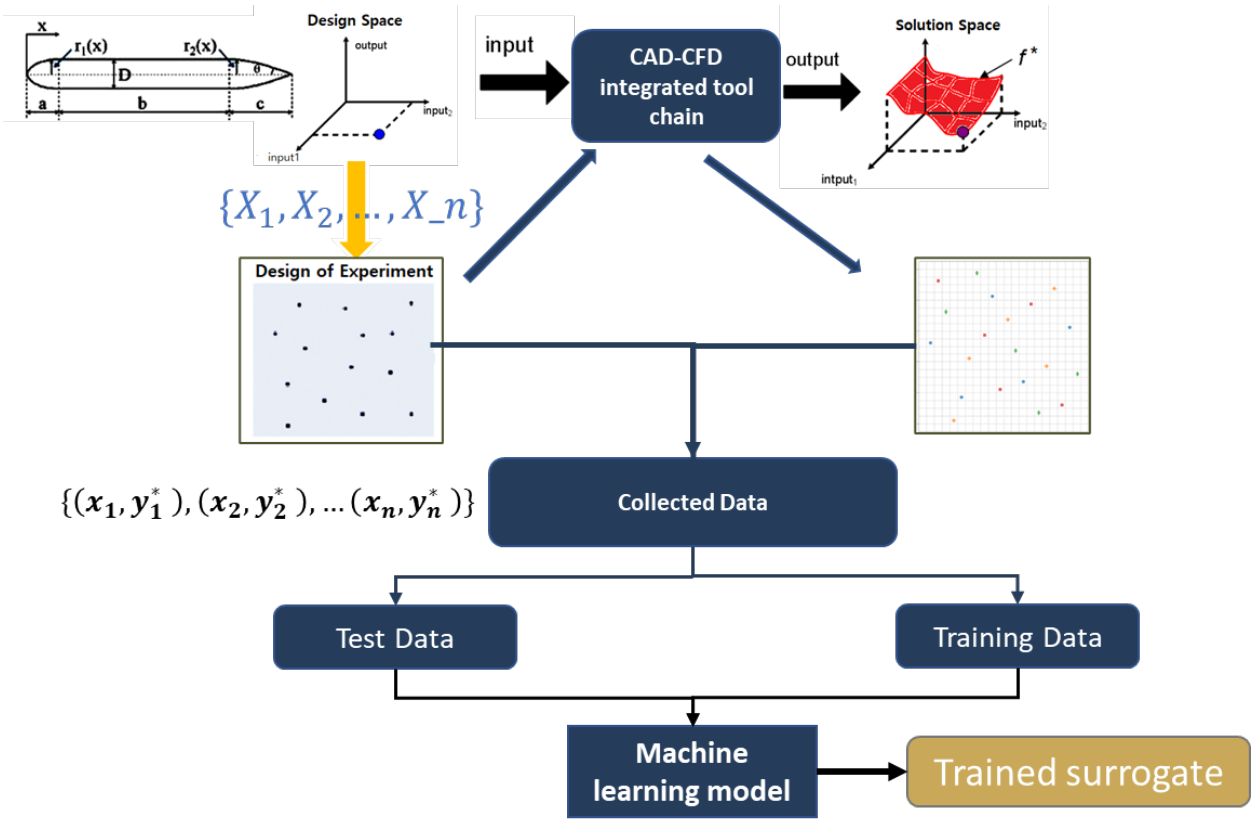


Figure 11. Surrogate-based design optimization for UUV.

3.2.2.4 Surrogate-based design optimization

Physics-based simulations present a significant computational hurdle in the optimization of computer-aided design for engineering tasks. Thus, to integrate precise yet computationally demanding simulations into the design optimization workflow, one must employ either a highly efficient optimization algorithm or develop swift surrogate models that can emulate the outcomes of lengthy simulations.

In our research, we embrace the latest advancements in both optimization and artificial intelligence to explore these avenues within the framework of optimizing the design of an unmanned underwater vehicle, as shown in Figure 11.

We begin by evaluating various optimization methods for their ability to efficiently use samples and their convergence trends when combined with a standard computational fluid dynamics solver. Following this, we craft a deep neural network that serves as a surrogate to estimate drag forces, which would typically require direct numerical simulations using the CFD solver. This surrogate is then integrated into the hull design optimization loop.

Our findings indicate that the Bayesian Optimization with Lower Condition Bound (BO-LCB) approach outperforms other methods in terms of sample efficiency and convergence.

Furthermore, our DNN surrogate model closely predicts drag forces, aligning with results from CFD simulations with a mean absolute percentage error (MAPE) of just 1.85%. By leveraging

these insights, we achieve a remarkable hundredfold increase in the speed of the design optimization process, while maintaining accuracy utilizing the surrogate model.

To the best of our knowledge, this study is first of its type that applies Bayesian optimization and DNN surrogates to UUV design optimization. We contribute to the community by making our tools available as open-source software.

3.2.2.5 Surrogate Modeling using Physics-guided Learning

Computer simulation models are used extensively in scientific and engineering problems for complex design tasks and decision processes. Surrogate models generated using data-driven techniques can approximate the behavior of complex simulation models with high fidelity and can accelerate the design process. We developed a physics-guided learning architecture, see Figure 12, that integrates parameters extracted from physics-based simulations into the intermediate layers of a neural network to constrain the learning process during the training of surrogate models and to improve their generalization. The architecture is used to develop a surrogate model for evaluating the structural integrity of the hull of an unmanned underwater vehicle. It is shown that physics-guided learning can improve generalization in less explored regions of the design space compared to black-box models. In addition, the architecture improves the explainability of the model predictions using physics-based parameters, see Table 1, which allows the designer to make decisions based on the input and physics-based intermediate parameters.

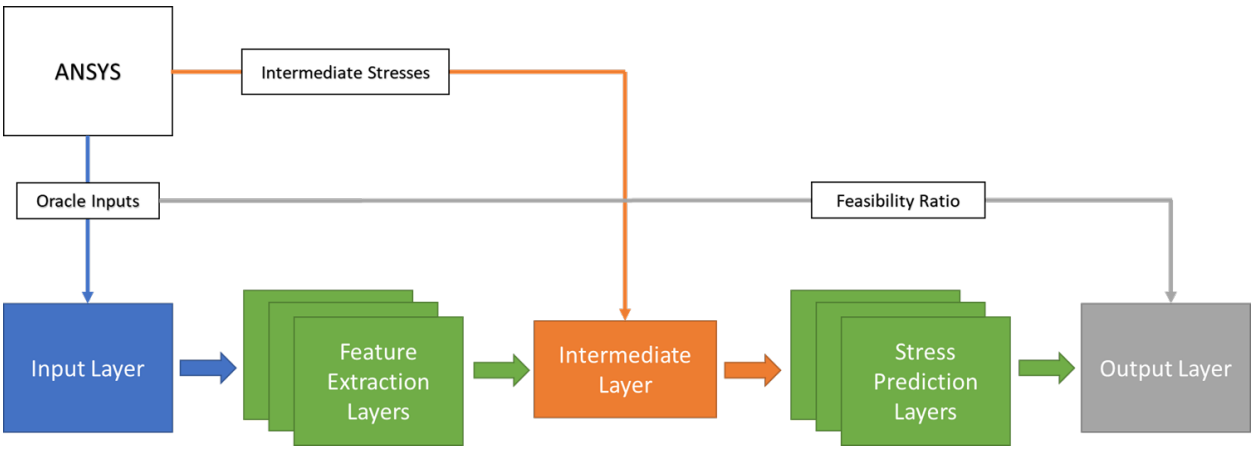


Figure 12. Surrogate Modeling using Physics-guided Learning.

Table 1: Example physics-based parameters for learning architecture.

Oracle Inputs		Simulator Output		
Property	Description	Property	Description	Notes:
E	Material Elasticity	f_r	feasibility ratio	σ_{max}/σ
σ	Material Strength		Intermediate Physics-based Variables	
υ	Poisson Ratio	$\sigma_{x,y,z}$	Maximum Directional Stresses	
ρ	Material Density	$\sigma_{xy,yz,xz}$	Maximum Plane Stresses	
r_i	Inner Diameter			
t	Thickness			
l	Cylinder Length			
σ_{hyd}	External Hydraulic Pressure			
n	Number of Stiffeners			
t_s	Thickness of Stiffener			
h_s	Height of Stiffener			

3.3 CIRCUIT MAKER - AI-ASSISTED ELECTRONIC DESIGN AUTOMATION

The ambitious goals of the circuit maker project were to work toward the synthesis of electronic circuits. This effort was split into smaller milestones, including curating a dataset of electronic circuits, autocompletion of electronic circuits, and developing a labeling tool to assist in the recognition of electronic circuits from schematic diagrams. The first milestone, curating a dataset of electronic circuits, focused on collecting public electronic circuits in the form of Simulation Program with Integrated Circuit Emphasis (SPICE) netlists. We started with netlists packaged with LTspice, including example and demo circuits then proceeded to scraping public circuits from GitHub. Basic statistics on these datasets were collected and are available at SPICE Netlist Dataset Metrics (symbench.github.io).

The second milestone used the data collected in the first milestone to explore the autocompletion of electronic circuits. We formalized the task as a graph classification task where we preprocessed the original dataset by removing a single component and using that as the class label. Then we explored various graph classification algorithms, including multiple graph neural network (GNN) architectures and using graph descriptors based on control theory. This work was published in Neural Computing and Applications. We also developed an electronic circuit design tool using Web-based Generic Modeling Environment (WebGME), a generic framework for developing domain-specific design tools, integrating this work to be able to evaluate the tool interactively.

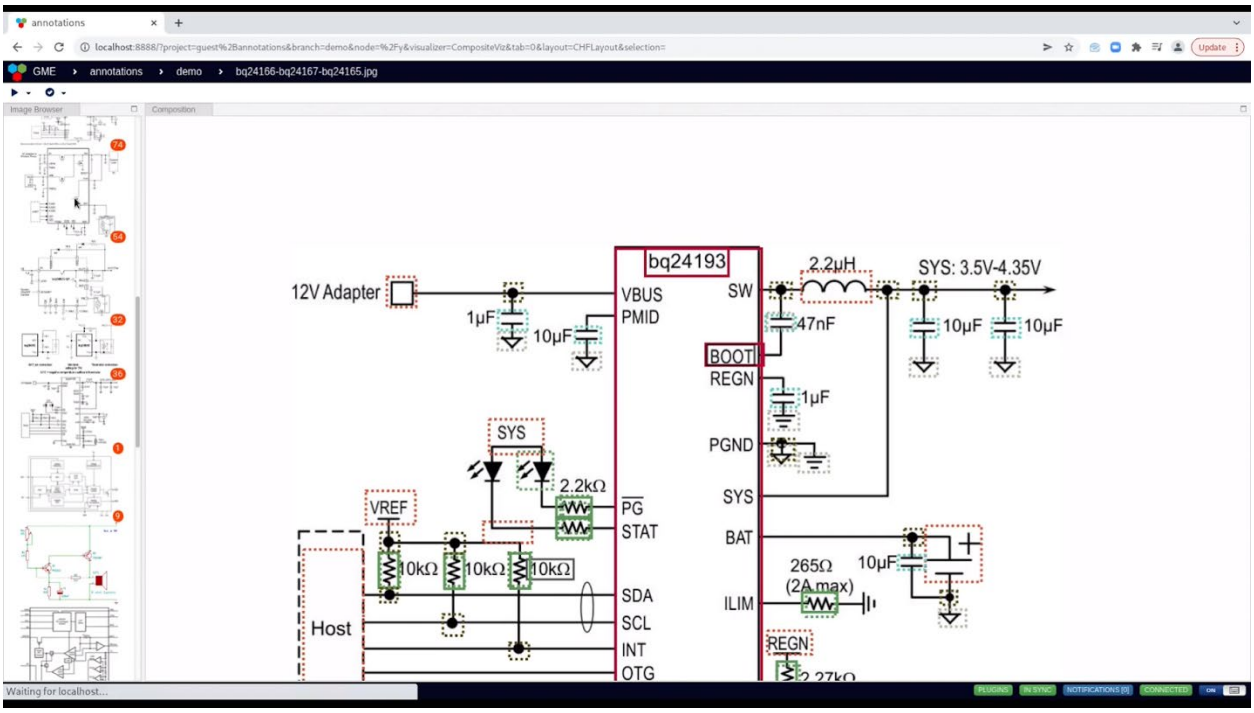


Figure 13. AI-assisted Electronic Circuit Recognition.

Our work on the second milestone exposed one shortcoming of the collected dataset: many of the circuits were either 1) toy examples or 2) low quality. Although there is no standard exchange format, datasheets for commercial components include highly optimized circuits in the form of a schematic diagram. To this end, our third milestone focused on the automatic conversion of these schematics to a meaningful, machine-readable representation. First, we scraped thousands of schematic diagrams from Texas Instruments. Then, we designed a structured representation to capture the knowledge conveyed in these schematics (as they were a superset of SPICE and included a lot of informal notations). Next, we developed a configurable labeling tool for the formally represented labels (using WebGME). This tool was designed to make the herculean task of labeling the schematic diagrams tractable for our small team and incorporated many AI tools to facilitate automation, including automatic detection of components within an image from a single label and integrated optical character recognition (OCR), see Figure 13.

Using the data collected from the labeling tool, we trained object detection models for detecting the components from images and began exploring how we can leverage machine teaching to integrate expert knowledge and purely statistical models into the program, shown in Figure 14. To this end, we were developing a compiler to create ensembles of intelligent models from a knowledge specification provided by a domain expert. This was a superset of the prior approach as training object detectors could be performed by simply specifying a component via a set of examples. In this case, a You Only Look Once (YOLO) model was trained as before. However, this approach was designed to enable designers to specify the task in a variety of other ways - as they may when conveying the knowledge to another person. These heuristics are then automatically ensembled alongside the statistical models to bootstrap the overall labeling models.

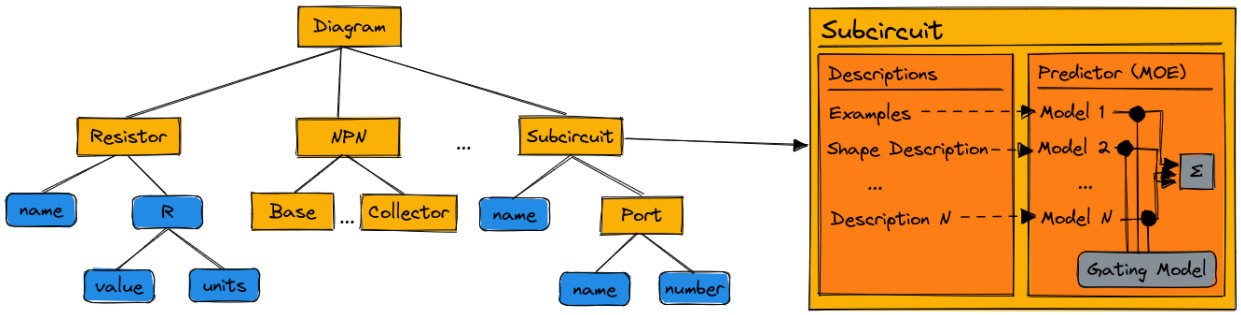


Figure 14. Circuit Classification with Machine Teaching.

We also demonstrated the feasibility of using graph neural networks for design completion tasks with circuit models, shown in Figure 15. Given an incomplete netlist, the goal was to predict the missing component(s) and links in the circuit. We transformed each circuit into a graph where nodes correspond to components and edges correspond to their connectivity. We then removed each component from the graph one by one, stored, and labeled the graph with the removed component.

We adopted a graph machine learning approach to predict the missing component. We applied both graph descriptors and graph neural networks and compared the results.

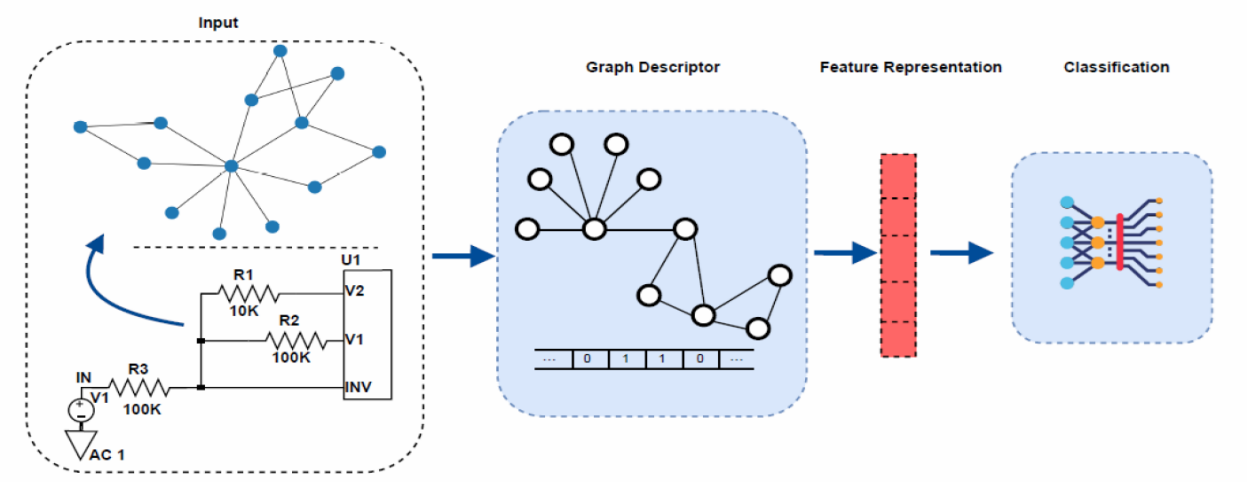


Figure 15. Graph Neural Network-based Circuit Completion.

4.0 RESULTS AND DISCUSSION

4.1 UAV / UAM Design Results

In both Hackathons and for the Phase 1 Demonstration, we managed to generate several airworthy vehicle designs (i.e., completing all flight paths). Indeed, all of our submitted AI-driven solutions outperformed the manually crafted technical area (TA) 3 seed designs.

In Hackathon 1, we were able to automate the component selection process only. By the second Hackathon, we managed to explore several distinct (and auto-generated) vehicle topologies. Our design submission received the second-best overall score (8271), see Table 2, among all TA 1 performers. The most significant result was to reach a stable 40 m/s trim state – something that seemed unattainable with human intuition or manual tweaking.

Table 2: UUV Design Performance Scores - Hackathon2.

Design	Interferences	Score		
		Loaded	Unloaded	Total
FalconSM4Rotated	0	2763	2751	8271
FalconS8RotatedCargoCase	0	2757	2746	8254.5
FalconS4RotatedCargoCase	0	2741	2735	8214
FalconS4	0	2727	2721	8172
FalconM4RotatedCargoCaseb	0	2650	2643	7939.5
FalconM4b	0	2636	2636	7908
UnoInlineUAV	0	2573	2616	7783.5
TIE4	0	2567	2566	7699.5
FalconM4	0	2560	2560	7680
FalconT8	0	2444	2457	4901
FalconLight	0	2395	2395	4790
TiltieTailed	0	2153	2158	4311

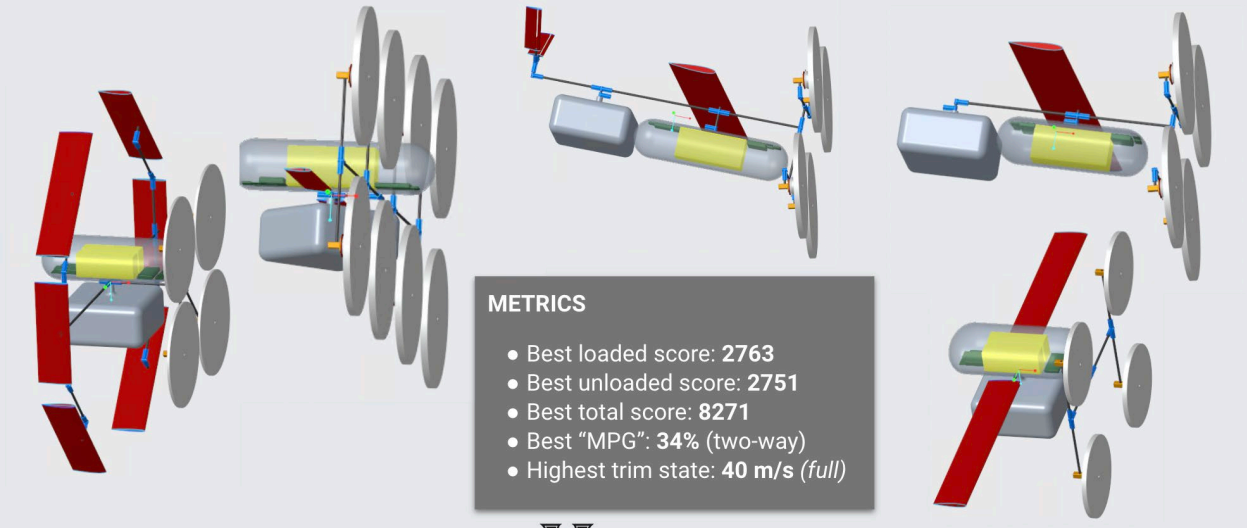


Figure 16. Final UUV Designs - Hackathon2.

Figure 16 shows the variability of the submitted final designs.

4.2 UUV Design Results

One of the most significant results in this design domain was the speed of design studies during Hackathons and at the end of the Phase 1 demo. By using SymCAD / SymDesign - and the underlying constraint programming toolbox, we managed to explore 140 alternative vehicle layouts and identify 19 feasible topologies, resulting in 9 unique parametric design points. Each parametric design point had ~100 sizing options. The final design was identified on the Pareto-front by human decision (trade-off). All of these steps required a few hours of execution time. Our end of Phase1 slides contain a short video of this exploration process. In the Hackathon2, the same approach demonstrated extremely fast (1 day, most of the time) adaption required for challenge perturbations to be captured as changed / additional mission requirements.

The Figure 17 and Figure 18 show the auto-generated CAD geometry at the end of Hackathon2, with Table 3 listing the UUV performance characteristics.

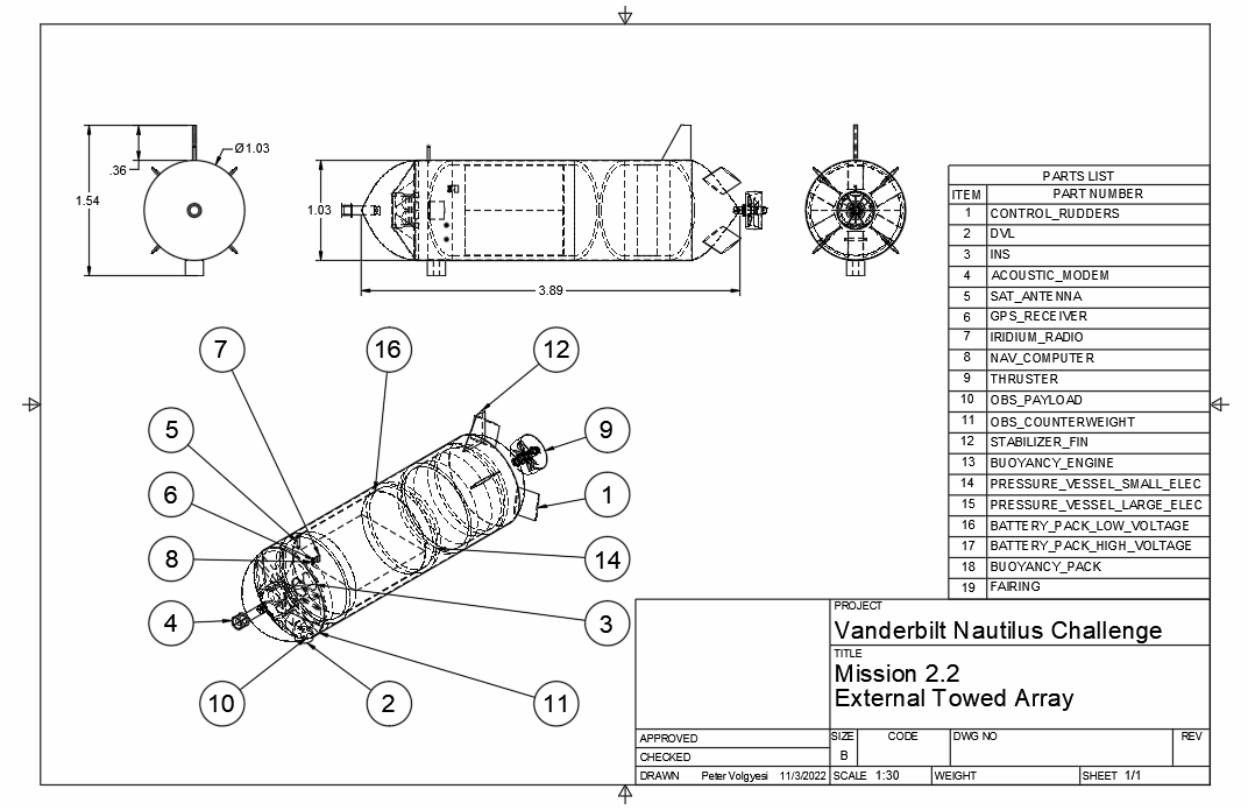


Figure 17. Design Drawing of UUV - Hackathon2.

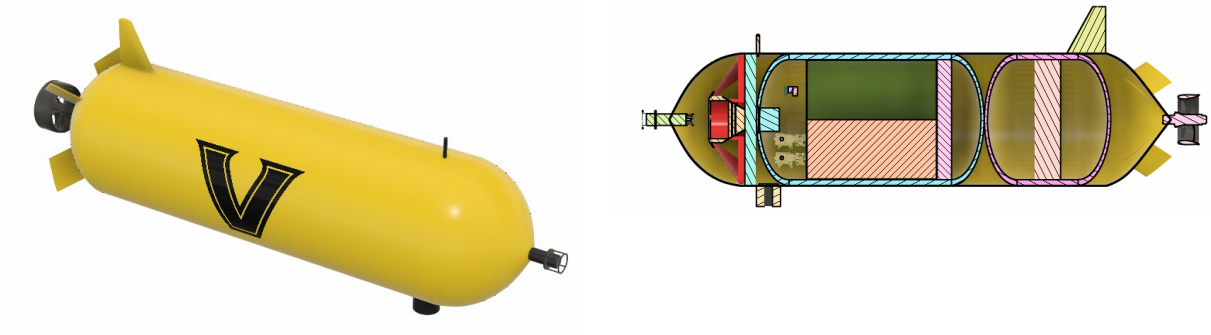


Figure 18. Auto-generated UUV CAD geometry – Hackathon2.

Table 3: UUV Performance Characteristics - Hackathon2.

	Dry Mass (kg)	Displacement (m^3)	Volume (m^3)	Length (m)	Min Pitch (°)	Max Pitch (°)	Propulsion Energy (kWh)
Baseline	1758	2.364	2.680	3.879	27.25	88.55	—
Internal Sensor Array	2099	2.941	3.222	4.534	87.66	86.37	110.77
Towed Sensor Array	1760	2.372	2.690	3.891	30.21	87.95	87.54

4.3 Circuit Completion – Missing Component Prediction

The results of various GNN and Graph Descriptor methods for classifying a missing component in a circuit model are shown in Table 4 and Table 5.

Table 4: Circuit Component Classification Results of GNN Methods (F1 and standard deviation).

Dataset	GCN	GIN	GAT	GraphSAGE	NGCN	NGIN	NGAT	NGraphSAGE
Ltspice Examples	79.5 ± 0.01	$\mathbf{89.8} \pm 0.01$	78.1 ± 0.04	78.6 ± 0.03	81.9 ± 1.04	88.9 ± 0.01	82.7 ± 0.03	82.2 ± 0.03
Ltspice Demos	61.8 ± 0.01	$\mathbf{85.0} \pm 0.02$	58.5 ± 0.03	59.0 ± 0.04	52.6 ± 0.08	82.4 ± 0.03	53.8 ± 0.09	53.7 ± 0.06
Kicad Github	51.7 ± 0.03	$\mathbf{67.4} \pm 0.02$	56.7 ± 0.04	46.6 ± 0.02	58.1 ± 0.01	62.5 ± 0.02	56.8 ± 0.03	57.0 ± 0.02

Table 5: Circuit Component Classification Results of Graph Descriptors and Kernel Methods (F1 and standard deviation).

Dataset	DGSD	NetLSD	WL	FGSD	NetSimile	Shortest Path
Ltspice Examples	77.63	77.22	77.21	48.88	**85.11**	77.21
Ltspice Demos	57.60	52.07	48.27	33.03	**70.86**	48.26
Kicad Github	47.89	47.77	49.35	51.52	**59.27**	49.35

5.0 CONCLUSIONS

One of the most fundamental challenges in developing AI-assisted tools and technologies for CPS design is the lack of curated, homogenous, machine-readable large datasets. The problem can be partially mitigated by synthetic models and simulator feedback, but this approach is highly resource intensive – and, in complex design cases, not scalable.

We identified the electric circuit design domain as one of the potential exceptions, where readily available (open source) design artifacts are available. Even in this domain, the sample size is in the thousands, which limits the training of really large and sophisticated AI models.

Therefore, to tackle the UAV / UAM / UUV design challenges successfully, we used a more heterogeneous design flow by applying AI and machine learning (ML) selectively. First and foremost, the conceptual design phase was primarily supported by a multi-objective constrained optimization approach. The relevance of AI in this step is the PyTorch library (primarily developed for ML applications), which enabled us to optimize and satisfy complex functions and constraints in a vectorized approach on multicore central processing unit (CPU) / graphics processing unit (GPU) hardware. The overall approach and the runtime libraries also enabled us to integrate trained NN and curve-fitted models in these computational graphs.

The most fitting area for AI / ML proved to be surrogate modeling, where these models were trained on synthetic / simulated data to learn some important characteristics (surfaces) of various analysis domains (CFD, FEA, geometric / packing, electromechanical). With a limited amount of computational resources, we managed to demonstrate this approach in multiple design domains, albeit with a limited set of degrees of freedom in the design parameter space.

One particular area where a large set of machine-readable complete designs would (have been) highly beneficial is in the development of AI-assisted tools for generating practical vehicle architectures (design topologies). Even with a grammar-based approach, our current method is highly limited and results in overwhelmingly infeasible / impractical designs.

Finally, the given set of analysis tools (e.g., SwRI / ATHENS Flight Dynamics Model) are extremely sensitive to small parameter changes, resulting in highly unpredictable error / cost / performance surfaces. Thus, our overall approach always relied on systematic or genetic algorithm (GA) based methods as a final optimization step.

6.0 REFERENCES

Kaneda, A., Akar, O., Chen, J., Kala, V.A.T., Hyde, D. & Teran, J.. (2023). "**A Deep Conjugate Direction Method for Iteratively Solving Linear Systems."** *Proceedings of the 40th International Conference on Machine Learning, PMLR (*accepted)

Carlos Olea, Michael Sandborn, Peter Volgyesi and Jules White, "**String Grammars for Preliminary UAV Design Exploration**", *2023 14th International Conference on Mechanical and Aerospace Engineering (ICMAE)*, Porto, Portugal, July 18-21, 2023 (presented)

Harsh Vardhan, David Hyde, Umesh Timalsina, Peter Volgyesi, Janos Sztipanovits, "**Sample-Efficient and Surrogate-Based Design Optimization of Underwater Vehicle Hull**", *Journal of Computational Physics*, (submitted 2023 Apr)

Harsh Vardhan, Peter Volgyesi, Will Hedgecock and Janos Sztipanovits: "**Constrained Bayesian optimization for Automatic Underwater vehicle hull design**", Design Automation for CPS and IoT (DESTION 2023), IEEE/ACM CPS-IoT WEEK, May 9, 2023, San Antonio, Texas, USA (submitted)

Ali Ozdagli, Peter Volgyesi and Xenofon Koutsoukos: "**Surrogate Modeling using Physics-guided Learning**", Design Automation for CPS and IoT (DESTION 2023), IEEE/ACM CPS-IoT WEEK, May 9, 2023, San Antonio, Texas, USA (submitted)

Anwar Said, Mudassir Shabbir, Brian Broll, Waseem Abbas, Peter Volgyesi, Xenofon Koutsoukos. "**Circuit Design Completion using Graph Neural Networks**", *Neural Computing and Applications*, 2023

Said, Anwar and Shabbir, Mudassir and Hassan, Saeed-Ul and Hassan, Zohair Raza and and Ahmed, Ammar and Koutsoukos, Xenofon, "**On Augmenting Topological Graph Representations for Attributed Graphs**", *in Elsevier Applied Soft Computing,* 2023.

Harsh Vardhan, Umesh Timalsina, Peter Volgyesi, Janos Sztipanovits: "**Data efficient surrogate modeling for engineering design: Ensemble-free batch mode deep active learning for regression**" , Nov 2022, Engineering Applications of Artificial Intelligence (submitted)

Michael Sandborn, Carlos Olea, Anwar Said, Mudassir Shabbir, Peter Volgyesi, Xenofon Koutsoukos, Jules White, "**What a drag! Streamlining the UAV design process with design grammars and drag surrogates**". 2022 International Conference on Computational Science and Computational Intelligence (CSCI), December 2022. (submitted)

Harsh Vardhan, Janos Sztipanovits, "**Deep Active Learning for Regression Using ε-weighted Hybrid Query Strategy**", ReALML, International Conference on Machine Learning (*ICML*), Baltimore, July 2022 (presented)

Harsh Vardhan, Janos Sztipanovits, "**Deep Learning-based FEA surrogate for sub-sea pressure vessel**", 2022 6th International Conference on Computer, Software, and Modeling (*ICCSM*), Rome, Italy, July, 2022 (presented)

M. Maroti, W. Hedgecock, P.Volgyesi. "**Rapid Design Space Exploration with Constraint Programming**", *The 4th Workshop on Design Automation for CPS and IoT (DESTION)*. May 2022. (presented)

A. I. Ozdagli, and X. Koutsoukos. **"Physics-guided Deep Learning with Domain Adaptation for Structural Health Monitoring**", *Computer-Aided Civil and Infrastructure Engineering*. (in review)

A. I. Ozdagli, and X. Koutsoukos. **"Model-based Damage Detection through Physics-guided Learning for Dynamic Systems**", *13th Annual Conference of the Prognostics and Health Management Society*. November 2021.

A. I. Ozdagli, and X. Koutsoukos. "**Domain Adaptation for Structural Health Monitoring**", Annual Conference of the PHM Society 2020. (presented)

A. I. Ozdagli, and X. Koutsoukos. "**Domain Adaptation for Structural Health Monitoring under Model Uncertainty**", International Journal of Prognostics and Health Management. 2021. https://doi.org/10.36001/ijphm.2021.v12i2.2948.

Harsh Vardhan, Janos Sztipanovits, "**Reduced Robust Random Cut Forest for Out-Of-Distribution detection in machine learning models**", under review in 2021 8th IEEE Intl. Conference on Soft Computing & Machine Intelligence (ISCMI 2021).

Abbas, Waseem and Shabbir, Mudassir and Said, Anwar and Ahmed, Ammar and Koutsoukos, Xenofon, "**Network Controllability Perspectives on Graph Representation**", *submitted to* International Conference on Learning Representations (submitted to TKDE 2023).

Harsh Vardhan, Peter Volgyesi, Janos Sztipanovits, "**Machine Learning Assisted Propeller Design**". *The 12th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS 2021)*. Nashville, TN, USA, May 19-21, 2021

Harsh Vardhan, Janos Sztipanovits, "**Rare event failure test case generation in Learning-Enabled-Controllers**". *2021 6th International Conference on Machine Learning Technologies (ICMLT 2021*). South Korea, April 23-25, 2021

Quchen Fu, Zhongwei Teng, Jules White, Douglas C. Schmidt, "**A Transformer-based Approach for Translating Natural Language to Bash Commands**". *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA).* DECEMBER 13-16, 2021, Pasadena, California

Quchen Fu, Zhongwei Teng, Jules White, Maria E. Powell, Douglas C. Schmidt, "**FastAudio: A Learnable Audio Front-End for Spoof Speech Detection**". *2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* May 22-27, 2022, Singapore

Teng Zhongwei, Fu Quchen, White Jules, Douglas C. Schmidt, "Sketch2Vis: **Generating Data Visualizations from Hand-drawn Sketches with Deep Learning**". 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA). DECEMBER 13-16, 2021, Pasadena, California

Award: 1st place at NeurIPS 2020 / NLC2CMD Challenge (Nov 21, 2020)

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

.csv	Comma Separated Value
3D	Three-Dimensional
AI	Artificial Intelligence
AL	Active Learning
API	Application Programming Interface
ATHENS	Air Taxi (Hybrid or Electric) aeroNautical Simulation
AVM	Adaptive Vehicle Make
BO	Bayesian Optimization
BO-LCB	Bayesian Optimization with Lower Condition Bound
CAD	Computer Aided Design
CFD	Computational Fluid Dynamics
CLI	Command-line Interface
CPS	Cyber-Physical Systems
CPU	Central Processing Unit
CREO	CREO Parametric, formerly Pro Engineer
DARPA	Defense Advanced Research Projects Agency
DC	Design Composition
DNN	Deep Neural Network
DOD	Department of Defense
DSC	Design Space Construction
DSE	Design Space Exploration
FEM	Finite Element Method
GA	Genetic Algorithm
GNN	Graph Neural Network
GPU	Graphics Processing Unit
JSON	JavaScript Object Notation
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
NAUTILUS	Not an acronym, honors the first trans-Artic under-ice crossing (USS Nautilus, 1958)
NN	Neural Network

NOAA	National Oceanic and Atmospheric Administration
OCR	Optical Character Recognition
OpenFOAM	Not an acronym
OpenMDAO	Open Multi-Disciplinary Design Analysis and Optimization
PDF	Portable Document Format
PROLOG	Programming in Logic
SDCPS	Symbiotic Design for Cyber Physical Systems
SPICE	Simulation Program with Integrated Circuit Emphasis
STEP	Standard for the Exchange of Product Data
STL	stereolithography, sometimes called Standard Triangle Language or Standard Tessellation Language
STR	Systems & Technology Research LLC
SwRI	Southwest Research Institute
TA	Technical Area
UAM	Urban Air Mobility
UAV	Unmanned Aerial Vehicle
UUV	Unmanned Underwater Vehicle
WebGME	Web-based Generic Modeling Environment
YAML	originally, Yet Another Markup Language, but overtime went to YAML Ain't Markup Language. It is a human-readable data serialization language
YOLO	You Only Look Once