AircraftVerse: A Large-Scale Multimodal Dataset of Aerial Vehicle Designs

Adam D. Cobb,* Anirban Roy,* Daniel Elenius,* F. Michael Heim,† Brian Swenson,‖

Sydney Whittington,‖ James D. Walker,† Theodore Bapty,‡ Joseph Hite,‡ Karthik Ramani,§

Christopher McComb ,¶ Susmit Jha*

*Computer Science Laboratory, SRI International
†Mechanical Engineering Division, Southwest Research Institute
‖Intelligent Systems Division, Southwest Research Institute
‡Institute of Software Integrated Systems, Vanderbilt University
§School of Mechanical Engineering, Purdue University
¶Department of Mechanical Engineering, Carnegie Mellon University

Abstract

We present `AircraftVerse` a publicly available aerial vehicle design dataset. Aircraft design encompasses different physics domains and, hence, multiple modalities of representation. The evaluation of these cyber-physical system (CPS) designs requires the use of scientific analytical and simulation models ranging from computer-aided design tools for structural and manufacturing analysis, computational fluid dynamics tools for drag and lift computation, battery models for energy estimation, and simulation models for flight control and dynamics. `AircraftVerse` contains 27,714 diverse air vehicle designs - the largest corpus of engineering designs with this level of complexity. Each design comprises the following artifacts: a symbolic design tree describing topology, propulsion subsystem, battery subsystem, and other design details; a STandard for the Exchange of Product (STEP) model data; a 3D CAD design using stereolithographic (STL); a 3D point cloud for the shape of the design; and evaluation results from high fidelity state-of-the-art physics models that characterize performance metrics such as maximum flight distance and hover-time. We also present baseline surrogate models that use different modalities of design representation to predict design performance metrics, which we provide as part of our dataset release. Finally, we discuss the potential impact of this dataset on the use of learning in aircraft design and, more generally, in CPS. `AircraftVerse` is accompanied by a data card, and it is released under Creative Commons Attribution-ShareAlike (CC BY-SA) license. The dataset is hosted at `https://zenodo.org/record/6525446`, baseline models and code at `https://github.com/SRI-CSL/AircraftVerse`, and the dataset description at `https://aircraftverse.onrender.com/`.

1 Introduction

Publicly available datasets of complex cyber-physical systems (CPS) are difficult to build. Their availability is limited for multiple reasons, such as proprietary restrictions, difficulty in assembling the right mix of experts, and slow manual design process. However, a huge opportunity exists to apply data-driven approaches to CPS once such a dataset becomes widely available. Electric Vertical Take-Off and Landing (eVTOL) aircrafts represent an emerging class of CPS. The use of electrical propulsion and the growing energy density of available batteries have fueled rapid growth in eVTOL aircraft designs from food delivery/logistics [12], to the detection of sharks along highly populated beaches [18], and to air taxis. We expect this variation to only increase as battery technology continues to improve [10]. The diversity within the electric aerial vehicle design space is large due to the many choices available for selecting structural, mechanical, and electrical components. The multiphysics nature of CPS designs necessitate heterogeneity in representation, such as the use of stereolithographic (STL) files for computational fluid dynamics (CFD) analysis and symbolic description with description of motors and speed controllers for electrical analysis. Data-driven learning methods for CPS must be able to handle this diversity and multimodality of CPS designs. `AircraftVerse` provides such a dataset to enable further research into data-driven methods for characterizing or designing CPS.
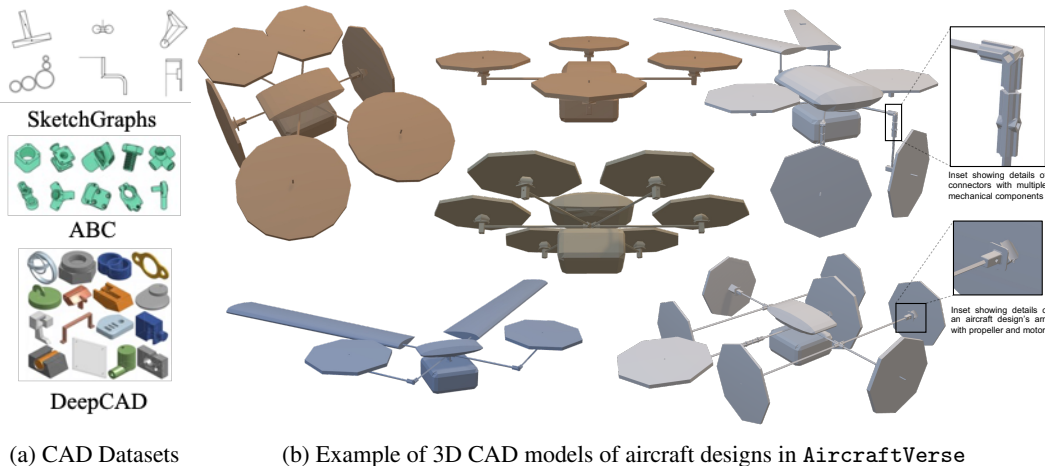
(a) CAD Datasets (b) Example of 3D CAD models of aircraft designs in `AircraftVerse`

Figure 1: Existing CAD datasets (SketchGraphs [27], DeepCAD [34], ABC [19]) are focused on CAD for mechanical parts. Aircraft designs in `AircraftVerse` include CAD models as one of the modalities. A CAD model (STeP or STL) is an assembly of several components such as propellers, wings, connectors, beams, motors, batteries, and hubs. The CAD designs in `AircraftVerse` are more complex compared to the existing datasets. The inset magnifications of a couple of parts of one of the designs in the figure above demonstrates this complexity. Further, in addition to CAD design, each design also includes a design tree with additional details such as propulsion and battery subsystems, in a symbolic form that are needed for performance analysis (e.g. electrical analysis, flight dynamics analysis). `AircraftVerse` also contains the result from the evaluation of each design using high-fidelity scientific and engineering tools. Thus, `AircraftVerse` is a CPS dataset and not just a CAD dataset.

`AircraftVerse`(Figure 1) contains **27,714 diverse aircraft designs** where each design is represented in multiple modalities, including CAD model and symbolic description. Each design is also accompanied with detailed evaluation results. Thus, `AircraftVerse`is naturally amenable to emerging neurosymbolic and other deep learning methods for generative modeling, surrogate learning and sequence-to-sequence models. Overall, our paper is structured as follows. In Section 2 we provide a summary of the few existing related datasets. We then introduce `AircraftVerse` in Section 1b and highlight the diversity of the dataset in Section 4. Finally, the potential of `AircraftVerse` in Section 4 by displaying experiments on surrogate modeling, noting that this covers just a small part of what kind of experimentation is possible with this new dataset. We then conclude in Section 5.

2 Related Work

The use of machine learning in computer aided design (CAD) has gained significant attention, and a few datasets have been proposed in recent literature to enable development and benchmarking of machine learning approaches. SketchGraphs dataset [27] is a collection of sketches extracted from parametric CAD models which begin as two-dimensional (2D) sketches consisting of geometric primitives (e.g., line segments, arcs) and explicit constraints between them (e.g., coincidence, perpendicularity) that form the basis for three-dimensional (3D) construction operations. This dataset has been used for generative model of CAD sketches [33], and other applications of learning in physical design [28, 23]. Another example of a CAD dataset that is focused on physical structure is SimJEB [32], which is a dataset of crowdsourced mechanical brackets and accompanying structural simulations. DeepCAD [34] is a dataset of 3D shapes corresponding to objects such as flanges, pipes and screws, represented as a sequence of operations used in a CAD framework to generate these shapes. Another dataset for 3D engineering shapes is the ABC dataset [19], which comprises geometric models, each defined by parametric surfaces and associated with ground truth information on the decomposition into patches. These datasets are excellent resources for their target application domains such as extrapolating 2D sketch to CAD designs, and generating mechanical parts.

In contrast, `AircraftVerse` is a dataset over the more complex design space of electric aircraft focused on system-level CPS design and thus, complements existing CAD datasets. In Figure 1, we illustrate the complexity of the aircraft designs in `AircraftVerse` compared to mechanical components from the existing CAD datasets. The CAD models of a aircraft design in `AircraftVerse` are an assembly of several mechanical components, such as propellers, motors, wings, hub and connectors. The insets in Figure 1 illustrate how parts of the design are composed of a number of components. Further, the `AircraftVerse` dataset includes additional description of a design beyond its CAD model, such as its propulsion subsystem and its electrical subsystem. This is crucial for creating a description that can predict the performance (flight dynamics, electrical analysis) of a design as designs with similar CAD structure can have very different performances depending on the used components. We also provide the performance of each design using detailed high-fidelity physics and engineering simulation tools to enable the use of this dataset not just for generative modeling, but also for learning surrogates and design characterization and optimization.

3 `AircraftVerse` Dataset: Structure and Curation

We now summarize the key characteristics of `AircraftVerse`:

- The number of designs in `AircraftVerse` is 27,714 making it a uniquely **large-scale CPS dataset** of with high design complexity. Design curation required finding valid aircraft configurations followed by detailed simulations to evaluate its performance objectives. We have attempted to create a balanced dataset that includes a sufficient number of aircraft designs across a range of different flight performance metrics, such as hover times and maximum flight distances. Our search for designs included over a hundred thousand candidates from which we selected these 27,714 designs to ensure diversity in design and performance. Our design process [5, 6] itself is diversity-preserving and ensures optimization does not lead to very similar designs.

- `AircraftVerse` represents design using **multiple modalities** that include: a symbolic design tree describing the design topology, propulsion subsystem, battery subsystem, and other design details; a STandard for the Exchange of Product (STEP) model that is a decomposable CAD model showing each part separately; and a stereolithographic (STL) CAD model that is ideal for computational fluid dynamics analysis, and the corresponding 3D point cloud for the shape of the design.

- `AircraftVerse` uses high-fidelity scientific models and evaluation pipeline developed as a part of DARPA's Symbiotic Design of CyberPhysical System's program[1] to evaluate different aircraft designs, and includes the **results from high-fidelity physics models** that characterize performance

---

[1] https://www.darpa.mil/program/symbiotic-design-for-cyber-physical-systems

metrics such as maximum flight distance and hover-time. The evaluation pipeline uses a mixture of custom flight dynamics simulators [30] and commercial tools such as Creo [24]. The evaluation of a design to determine performances such as its drag, lift, maximum flight time and hover time can take hours and requires subject-matter expertise. We include these evaluation results as part of the dataset.

- The designs in `AircraftVerse` exhibit a very **high degree of diversity** in their topology, the choice of energy subsystem and the choice of propulsion. The designs are a mixture of rotorcrafts, winged aircrafts and hybrids with 28% capable of vertical takeoff and landing.[2] To the best of our knowledge, there is no other available corpus of such diverse aircraft designs with the design details and the results from detailed scientific and engineering evaluation.

- We include a **baseline surrogate model** to predict some of the key design performance metrics. Our surrogate models include a transformer encoder model that takes the symbolic design configuration, as its input, a graph convolution neural network that uses the point cloud modality and an LSTM model. These baseline models are used to predict the mass, the maximum flight distance, the maximum hover-time and the presence of any structural interferences that needs to be avoided for fabrication and manufacturing.

Dataset Availability. The dataset is publicly available for free under Creative Commons Attribution-ShareAlike (CC BY-SA) license, which will enable future extension and adaptation of the dataset by others. With respect to its maintainability and long-term availability, the dataset is hosted at `https://zenodo.org/record/6525446` and will be maintained by SRI International[3], a non-profit research institute with over 75 years of history of contributing to society and research community, with a proven track record of building, maintaining and distributing several datasets such as BioCyc [16], Voices [26], and open-source tools[4] - some of which have been maintained for multiple decades.

Dataset Curation. The designs in `AircraftVerse` are battery-powered, where the propulsion comes from the electric motors that power the propellers. Our design corpus includes a range of propellers, motors, batteries, and fixed wings. The list of components is provided in Appendix F. The combination of wings and propellers (in the right topology) provides the thrust and lift needed for efficient horizontal flight and vertical take-off and landing. Our components are diverse, such as the propellers have a different number of rotor blades with different diameters and pitches, the component motors have different Kv, Km, Kt ratings, and the batteries have different peak and continuous current ratings and different voltage ratings. Consequently, the design choices allow significant diversity in structure and composition for similar performance. For example, the thrust produced by a propeller-motor pair can be related approximately to the parameters by the following formula:

$$Thrust \propto (K_V \times Voltage)/(Pitch \times Diameter)$$

where K_V is the motor's "Kilovolt" rating, which represents the number of revolutions per minute (RPM) that the motor will produce per volt, $Voltage$ is the voltage being applied to the motor, $Pitch$ is the distance that the propeller would move forward in one rotation, and $Diameter$ is the distance across the propeller. Our design used more detailed models [30, 24], but this dependency illustrates how the same thrust can be produced by different combinations of propellers and motors. High K_V rating motors that rotate much faster can produce the same thrust with smaller propellers when compared to lower K_V (slower) motors with larger diameter propellers. These combinations can have different continuous and peak current requirements that would influence the battery choice. In addition to these propulsion choices, connectors such as parametric joints, hubs, and arms can be put together in flexible ways to build interesting design geometry and topologies. These can have an impact on the experienced drag, which would then influence the requirements for propulsion. Thus, the design of aircraft requires making a number of tightly coupled design choices. The performance of each design can be assessed using a pipeline comprising commercial tools such as Creo [24] and custom

---

[2]We note that some aerial vehicles, such as many fixed wing aircraft, are launched (or catapulted) and therefore are not required to hover.

[3]https://en.wikipedia.org/wiki/SRI_International

[4]https://github.com/SRI-CSL

flight dynamics model (FDM) [30, 2]. Each aircraft is also assessed on controllability (existence of trim states - where the aircraft remains stable in the absence of environment perturbations) at different speeds using the FDM. In particular, the FDM evaluates whether the aerial vehicle can fly at a specific velocity through optimization schemes that check if the translational and rotational accelerations can be driven to zero for a given design during vertical take-off and horizontal flight (more details in Appendix C). Manually finding a large number of feasible designs is very challenging and prohibitively time-consuming. Instead, we use a learning-based design approach [5, 6]. We create a number of designs using a procedural grammar generator that uses heuristics provided by domain experts, such as limiting the complexity of produced designs to have at most 16 propellers and at most 12 wings, using design motifs and symmetries that help with controllability of the aircraft. A transformer-based model is trained on designs with good flight characteristics to act as a filter in the future generation process (Appendix E,[6]). Each of the final 27,714 designs included in the dataset is run through the detailed scientific and engineering models [30, 2, 24] to generate the metadata describing the design performance and characteristics.

Design Structure and Representation. Each design in the dataset consists of the following:[5]

- `design_tree.json`: The design tree describes the design topology, choice of propulsion and energy subsystems. The tree also contains continuous parameters such as wing span, wing chord, and the lengths of propeller arms and connectors. In our dataset, we also include a preorder traversal of the design tree and store this as `design_seq.json` to facilitate the use of sequence-friendly model architectures. In addition, we include `design_low_level.json`, which is a more fine-grained engineering representation of the design. This fine-grained representation includes significant repetition that is avoided in the abstract tree representation through the use of symmetry (such as specifying that the same arm structure is repeated six times around a hub). See the pictorial representation of the design tree in Figure 2 that demonstrates a simple example of such a symmetry.

- `Geom.stp`, `cadfile.stl`, and `pointCloud.npy`: CAD design for the aerial vehicle in compositional STP format (ISO 10303 standard), its stereolithographic STL file, and a generated pointcloud of 10,000 points from the CAD representation.

- `output.json` and `trims.npy`: Summary files containing the vehicle's performance metrics such as maximum flight distance, maximum hover time, fight distance at maximum speed, maximum current draw, and mass. The `trims.npy` contains the [*Distance, Flight Time, Pitch, Control Input, Thrust, Lift, Drag, Current, Power*] at each evaluated trim state (velocity) of the aircraft. Designs that have more trim states and with contiguous trim states are preferable as these designs would be relatively easier to control even when the environment is noisy. Thus, both the `output.json` and `trims.npy` contain the evaluation of the performance of each design. See Figure 2 for an example of these evaluation indicators.

Design Example. The design tree used for aircraft designs reflect the hierarchical nature of CPS designs. Figure 2 shows an example design. It has a *4-way central hub* component. Each of the four arms of the hub can have a propulsion subsystem attached to it. We identify this subsystem as a *MainSegment* which can be implemented via different choices (subtrees expanding *MainSegment*) such as *BendSegment*, *PropArm*, *ExtendedPropArm*, *WingArm*, and so on. Each of these choices describe the nature of the propulsion subsystem, such as whether we are using propeller or wing, and whether the arm making connection is a simple connector or a composition of connectors. Further, some of these subsystems can be recursive, allowing fractal growth of the design. For example, an arm itself might contain another segment with a hub having a number of arms. Our symbolic representation of design also enables exploitation of symmetry to compress the design description, and inclusion of expert knowledge in the design. For example, if we want all four arms of a hub to contain the same subsystem, then we only need to specify it once (as a single child in the design tree) and use a symmetry tag over the hub, for example, *ConnectedHub4_Sym* denotes a hub with four connections - each having the same subsystem. In the case of Figure 2, a propulsion subsystem comprises a propeller, motor, and a flange. Further, the hub also connects to a fuselage that contains
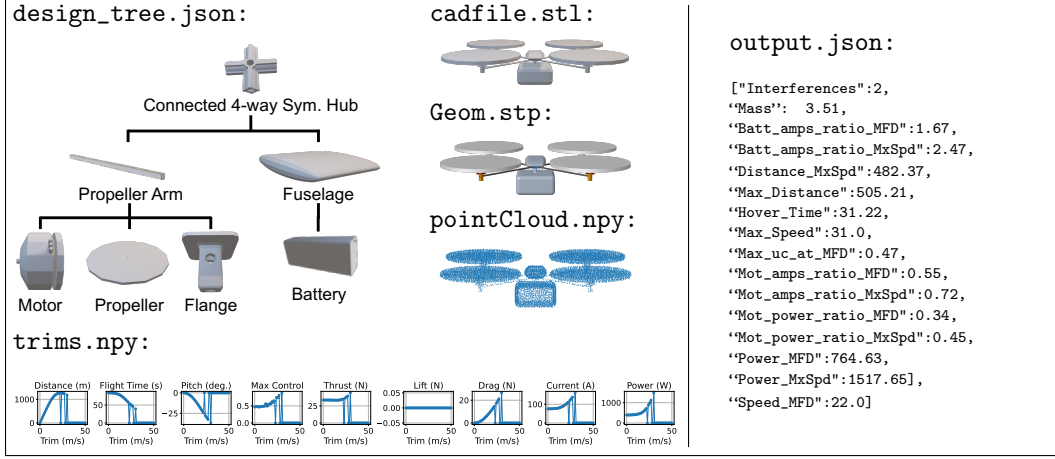
---

[5]For a complete printout see Appendix G.

Figure 2: An illustration of all the key file components that make up an aircraft design. The `design_tree.json` captures all aspects of the design (e.g. 4-way hub) that appears in both the `cadfile.stl` and the `Geom.stp`. Additional files of `design_seq.json` and `design_low_level.json` are directly derived from the design tree. The performance of the aerial vehicle is displayed in both the performance file labeled `output.json` and in the `trims.npy`. For example, here there is zero lift at all trims since the design has no wings.

a battery subsystem that can consist of a single or dual battery subsystem along with additional electronics and their position within the fuselage. An explicit example of this is provided in Appendix G. Additionally, the full corpus of options as included in Appendix F, is also provided as a Python dictionary as part of `AircraftVerse`.

Design evaluation. For each design, we use state-of-the-art tools [24, 2] to compute the physical characteristics, such as the mass and the component interferences, as well as to render the 3D CAD models (STP and STL files that are included in `AircraftVerse`). These details are then passed to the FDM [30] for evaluating flight dynamics (more details in Appendix C). The dynamics model considers the aircraft velocity and its roll, pitch and yaw rotations, along with the electrical state such as the power drawn from the battery and motor current, which in turn determine the torque on the propeller and its generated thrust. The dynamics model summarizes the impact of different forces such as gravity, drag, propeller lift and wing lift, and includes scientific models for conversion of electrical energy into mechanical energy in the propulsion subsystem. From these evaluations, we include *Batt_amps_ratio_MFD, Batt_amps_ratio_MxSpd* that correspond to the ratio of maximum current drawn by the design compared to the maximum current rating of the battery at maximum flight distance and maximum speed, respectively. These indicate the robustness of the battery subsystem of the design. We also include *Mot_amps_ratio_MFD, Mot_amps_ratio_MxSpd, Mot_power_ratio_MFD, Mot_power_ratio_MxSpd* that represent the ratio of the current/power in the motor divided by the maximum allowed current/power corresponding to the maximum flight distance or maximum speed of the vehicle. These indicate the robustness of the propulsion subsystem of the design, its endurance, and its agility. Finally, we include design metrics such as the flight distance at maximum speed (*Distance_MxSpd*), maximum flight distance (*Max_Distance*), and the maximum hover (*Hover_Time*).

4 Experiments: Diversity, Multimodality and Baseline Models

Diversity. `AircraftVerse` includes a diverse set of aircraft designs (samples shown in Figure 3), and the multimodal representation makes this dataset relevant to a variety of challenge problems in machine learning. The components of a design in `AircraftVerse` are selected from a large component corpus with hundreds of choices of propellers and motors (Appendix F). Each component also has a set of discrete and continuous parameters associated with it. The nodes in the design

Figure 3: Diverse vehicle designs in `AircraftVerse` with different structures and with different performances. Some designs can hover in place. Hover time is in seconds and flight distance is in meters.
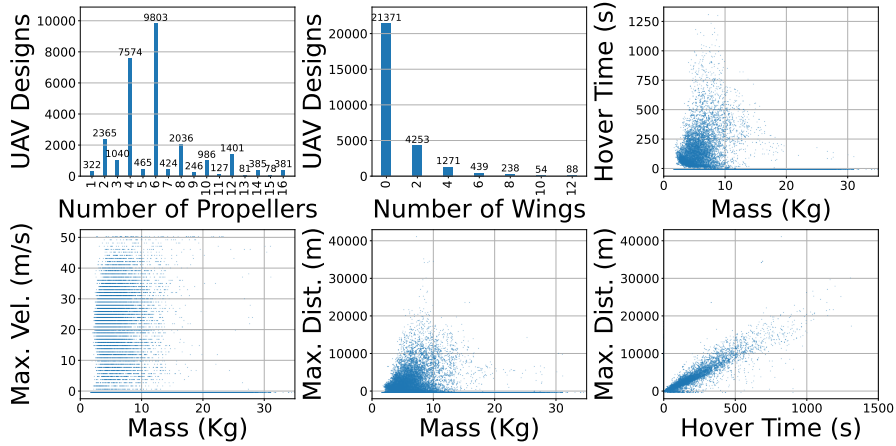

Figure 4: The designs in `AircraftVerse` exhibit diverse performance with respect to characteristics such as maximum flight distance and hover-time. They also have very different physical characteristics such as mass, number of propellers, and number of wings.

tree identify the subsystem or component, along with attributes corresponding to these parameters. For example, structural components such as connectors have their radius and length as continuous parameters, and functional components such as motor have a large set of parameters such as motor armature-winding resistance and inductance, mass, back e.m.f. constant, and motor torque constant. In Appendix F, we identify the attributes of the key components in an aerial vehicle design. The values of these attributes have been selected to enable designs with interesting structures and behaviors. Some characteristics of the design diversity are captured in Figure 4. The figure highlights some of our curation choices. First, the majority of designs have an even number of propellers due to the symmetry built into our tree structure definition. Second, our dataset precludes designs without propellers and favors designs with fewer wings. Finally, while a significant number of designs can take off and achieve horizontal flight (with a large variation in ability), one of the challenges in building a diverse dataset comes at the expense of many designs failing to have the hovering capability

7

or other eVTOL flight performance metrics. However, these designs are useful for learning potential rules as to why these designs have poor characteristics compared to others, so we also include these in the dataset.

While structural differences lead to visually striking diversity, we highlight here that design diversity has other aspects, such as the use of different components within the same design topology. Vehicle designs with the same topology can perform very differently. Figure 5 contains four symmetric quadcopters with the same topology (same design tree structure). However, their performance metrics differ significantly, with the maximum flight distance differing by a factor of over 2x. Additionally, the electronics of the quadcopter on the right varies from the other three in that it has two batteries rather than one. Thus, the performance of designs with similar topologies cannot be predicted from just from the CAD design, but requires the symbolic design description in `design_tree.json`.

Hover Time (s): 31.2 Flight Dist. (m): 505.2	Hover Time (s): 67.5 Flight Dist. (m): 1329.8	Hover Time (s): 83.9 Flight Dist. (m): 1268.4	Hover Time (s): 56.2 Flight Dist. (m): 880.2
Propeller: 19x6 Motor: AT4125KV540 Battery: 1000mAh 4S 75C	Propeller: 9x6 Motor: KDE600XF-1100-G3 Battery: 2200mAh 4S 75C	Propeller: 18x8 Motor: AT3530KV580 Battery: 2200mAh 3S 75C	Propeller: 16x4 Motor: AT4120KV500 Battery: 2x 1000mAh 4S 75C

Figure 5: Diverse aerial vehicle designs in `AircraftVerse` with same topology structure but different motor/propeller/battery choices leading to diverse design performance.

Multimodality. The `AircraftVerse` uses multiple modalities to represent an aircraft design, as illustrated earlier in Figure 2. This multi-modality of `AircraftVerse` makes it applicable to a wide class of machine learning models, and we mention a few where this dataset is relevant:

- Structured sequential data models: The success of transformer-based sequence models used in natural language [29, 9, 25, 3, 21] has driven their adoption and extension to structured data such as computer programs. The sequential data from the design trees in `AircraftVerse` would be an interesting further use case with significant non-local dependencies. Further, models incorporating physical and electrical constraints on the design components would also be an interesting avenue when working with `AircraftVerse`.

- 3D CAD models: There is a growing interest in using machine learning for CAD design [27, 33, 23, 32, 34, 19]. The 3D CAD models (STL and STP files) in `AircraftVerse` provide a large dataset with complex 3D shapes. The metadata associated with the designs can be used to train and evaluate methods that predict the physics metrics from the CAD models.

- 3D Pointcloud: The use of machine learning for learning generative models or manipulating or segmenting 3D point clouds has also received significant attention. `AircraftVerse` includes 3D point clouds extracted from the STL and STP files. Thus, this dataset with the pointclouds can be used as a benchmark for 3D point cloud classification (e.g., presence or absence of interference) [35, 4, 13] and regression (e.g., predicting flight characteristics).

- Multimodal: As highlighted earlier, CPS such as aircraft consist of multiple modalities. Therefore we would not expect the performance of electric aerial vehicle designs to be accurately predicted on the basis of only one of the modalities (e.g. the 3D CAD model). Recent work on vision-language models [8, 15, 1, 20, 7] would be especially relevant to large CPS datasets such as `AircraftVerse`. However unlike in the vision-language scenario, `AircraftVerse` has stricter metadata that enforces physical and electrical constraints.

Baseline Models. We describe a multimodal surrogate model for `AircraftVerse` as an example application. We use a transformer encoder [29] model and a LSTM [14] model to predict a set of design performance characteristics of an aircraft from its symbolic design tree description. In our example, even using these symbolic descriptions of a design contain information from multiple modalities, as these sequences include electrical, mechanical, and topological information. We also

compare the predictive performance applied to point cloud data as a comparison. For transformers, we use the design representation in `design_seq.json` and use our custom-built tokenizer to convert each token of a design sequence from the symbolic representation (e.g. {'node_type': 'Connect-edHub3_2_1'}) into a tensor. For our embedding, we assign one-hot encoding for the keys and an additional one-hot encoding for the values. Furthermore, for keys that require floats as values (e.g. 'armLength') we ensure that there is a class that corresponds to the float in the value embedding, as well as appending the value of the float to the tensor. Each token of the sequence is a 749 vector corresponding to the concatenation of 43 classes of keys, 673 classes of values, 32 possible attributes of electrical/mechanical components, and a final dimension for float values. We provide our tokenization code as part of the dataset release, as well as the baseline models. Further architectural details of the models are presented in Appendix D.

Results. The results of predicting the performance statistics from the design sequences is presented in Table 1. This table shows that the sequential design description can be used to predict the performance characteristics of a UAV design. Moreover, certain metrics such as the ability of a design to fly are unsurprisingly easier to predict, compared to how far a design is actually able to fly. For such statistics, incorporating 3D information, such as pointcloud data may be necessary and the `AircraftVerse` dataset includes these. Figure 6 shows both the receiver operator characteristic (ROC) curve and precision-recall curve for the two models.

(a) ROC curve   (b) Precision-Recall
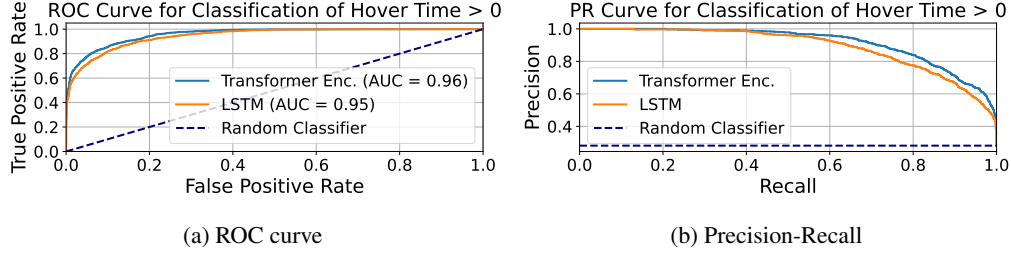
Figure 6: ROC curves and precision-recall curves for both the transformer encoder and the LSTM models when classifying whether a design would be able to hover. These results suggest that the symbolic design description in `AircraftVerse` can be used to predict flight characteristics. We also see that the transformer model does better than the LSTM model for this task with the same data.

Table 1: Specification prediction from sequences.

Approach	Data	Hover (Acc. ↑)	Hover Time (MSE, s^2 ↓)	Mass (MSE, Kg^2 ↓)	Flight Distance (MSE, Km^2 ↓)	Interference (F1 Score ↑)
T. Enc.	Sequence	0.9004	4843.5	0.1086	2.379	0.8270
LSTM	Sequence	0.8838	5728.8	0.3083	5.693	0.8184
GCNN	Point Cloud	0.8380				0.9758

5   Conclusion

We have introduced a new multi-modal CPS dataset that will enable researchers to explore a new interdisciplinary area of application for machine learning. We emphasize the richness of `AircraftVerse` where we provide 27,714 aircraft designs, each with associated metadata summarizing the performance of the design. We highlight that our experiments and baseline models have only touched the surface of the different approaches and problems that can be explored with this dataset due to the availability of all the metadata associated with each design. In addition to the material presented here, we also include an extensive supplementary materials section as well as a website: `https://aircraftverse.onrender.com/`, a GitHub repo: `https://github.com/SRI-CSL/AircraftVerse/`, and the dataset release at `https://zenodo.org/record/6525446`.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35: 23716–23736, 2022.

[2] Theodore Bapty, Sydney Whittington, James D. Walker, Joseph Hite, Brian Swenson, Kathrine Owens, Fred Eisele, Jason Scott, and Robert Owens. Design Oracle for AI-Based CPS Design. *Design Automation for CPS and IoT (DESTION)*, 2022.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[5] Adam Cobb, Anirban Roy, Daniel Elenius, Kaushik Koneripalli, and Susmit Jha. On diverse system-level design using manifold learning and partial simulated annealing. *Proceedings of the Design Society*, 2:1541–1548, 2022.

[6] Adam D Cobb, Anirban Roy, Daniel Elenius, and Susmit Jha. Design of unmanned air vehicles using transformer surrogate models. *arXiv preprint arXiv:2211.08138*, 2022.

[7] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning. *arXiv preprint arXiv:2305.06500*, 2023.

[8] Samyak Datta, Karan Sikka, Anirban Roy, Karuna Ahuja, Devi Parikh, and Ajay Divakaran. Align2ground: Weakly supervised phrase grounding guided by image-caption alignment. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2601–2610, 2019.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] Özgür Dündar, Mesut Bilici, and Tarık Ünler. Design and performance analyses of a fixed wing battery vtol uav. *Engineering Science and Technology, an International Journal*, 23(5): 1182–1193, 2020.

[11] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[12] Qiuchen Gu, Tijun Fan, Fei Pan, and Chong Zhang. A vehicle-uav operation scheme for instant delivery. *Computers & Industrial Engineering*, 149:106809, 2020.

[13] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(12):4338–4364, 2020.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[15] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pages 4904–4916. PMLR, 2021.

[16] Peter D Karp, Richard Billington, Ron Caspi, Carol A Fulcher, Mario Latendresse, Anamika Kothari, Ingrid M Keseler, Markus Krummenacker, Peter E Midford, Quang Ong, et al. The biocyc collection of microbial genomes and metabolic pathways. *Briefings in bioinformatics*, 20(4):1085–1093, 2019.

[17] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[18] Jeremy J Kiszka, Johann Mourier, Kirk Gastrich, and Michael R Heithaus. Using unmanned aerial vehicles (uavs) to investigate shark and ray densities in a shallow coral lagoon. *Marine Ecology Progress Series*, 560:237–242, 2016.

[19] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9611, 2019.

[20] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[21] OpenAI. Gpt-4 system card, 2023. URL https://cdn.openai.com/papers/gpt-4-system-card.pdf.

[22] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *arXiv preprint cs/0409058*, 2004.

[23] Wamiq Para, Shariq Bhat, Paul Guerrero, Tom Kelly, Niloy Mitra, Leonidas J Guibas, and Peter Wonka. Sketchgen: Generating constrained cad sketches. *Advances in Neural Information Processing Systems*, 34, 2021.

[24] PTC. Creo CAD Software. https://www.ptc.com/en/products/creo, 2022.

[25] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[26] Colleen Richey, Maria A. Barrios, Zeb Armstrong, Chris Bartels, Horacio Franco, Martin Graciarena, Aaron Lawson, Mahesh Kumar Nandwana, Allen Stauffer, Julien van Hout, Paul Gamble, Jeff Hetherly, Cory Stephenson, and Karl Ni. Voices obscured in complex environmental settings (voices) corpus, 2018.

[27] Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P Adams. Sketchgraphs: A large-scale dataset for modeling relational geometry in computer-aided design. *arXiv preprint arXiv:2007.08506*, 2020.

[28] Ari Seff, Wenda Zhou, Nick Richardson, and Ryan P Adams. Vitruvion: A generative model of parametric cad sketches. *arXiv preprint arXiv:2109.14124*, 2021.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[30] James D. Walker, F. Michael Heim, Bapiraju Surampudi, Pablo Bueno, Alexander Carpenter, Sidney Chocron, Jon Cutshall, Richard Lammons, Theodore Bapty, Brian Swenson, and Sydney Whittington. A Flight Dynamics Model for Exploring the Distributed Electrical eVTOL Cyber Physical Design Space. *Design Automation for CPS and IoT (DESTION)*, 2022.

[31] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[32] Eamon Whalen, Azariah Beyene, and Caitlin Mueller. Simjeb: Simulated jet engine bracket dataset. In *Computer Graphics Forum*, volume 40, pages 9–17. Wiley Online Library, 2021.

[33] Karl DD Willis, Pradeep Kumar Jayaraman, Joseph G Lambourne, Hang Chu, and Yewen Pu. Engineering sketch generation for computer-aided design. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2105–2114, 2021.

[34] Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6772–6782, 2021.

[35] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
    (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
    (b) Did you describe the limitations of your work? [Yes] We refer to Section 4 for comments regarding enforcing diversity in the dataset and to Appendix C for details on the flight dynamics model.
    (c) Did you discuss any potential negative societal impacts of your work? [N/A]
    (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [N/A]

(b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments (e.g. for benchmarks)...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]

   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]

   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] We include benchmarks approaches to demonstrate potential application domains. Our focus was in building the dataset which took up the vast majority of our computational resources.

   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix E

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

   (a) If your work uses existing assets, did you cite the creators? [Yes]

   (b) Did you mention the license of the assets? [Yes]

   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Data Card

In this section we include the data card that follows the structure in Pang and Lee [22] as highlighted in Gebru et al. [11].

A.1 Motivation

For what purpose was the dataset created? The dataset was created to enable research on designing cyber physical systems. In particular, we wanted to see if it was possible to predict characteristics of aircraft designs, and be able to filter out poor performing designs before needing to use expensive scientific models. In order to achieve this, we needed to build a large representative dataset to train our models.

Who created the dataset? The dataset was created by the authors: Adam D. Cobb, Anirban Roy, Daniel Elenius, and Susmit Jha, at SRI International; Brian Swenson, Sydney Whittington, and James Walker at Southwest Research Institute; Theodore Bapty, and Joseph Hite at Vanderbilt University; Christopher McComb at Carnegie Mellon University; supported by DARPA under the Symbiotic Design for Cyber-Physical Systems program with contract FA8750-20-C-0002.

Who funded the creation of the dataset? DARPA under the Symbiotic Design for Cyber-Physical Systems program with contract FA8750-20-C-0002.

A.2 Composition

What do the instances that comprise the dataset represent? The instances are aerial vehicle designs as generated by us and evaluated on scientific models [2, 30]. We also provide metadata in the form of the corpus dictionary such that the attributes of the propellers, motors, batteries, and wings are available.

How many instances are there in total There are 27,714 instances in total.

Does the dataset contain all possible instances or is it a sample of instances from a larger set? The dataset is a sample of instances that were generated according to our grammar. As our generator includes both discrete and continuous parameters, it is not possible to capture all possible instances of the designs. Furthermore, designs are allowed to contain recursively generated components, such as arms, such that all possible topologies cannot be sampled. The plots in Figure 4 show the diversity in designs.

What does each instance consist of? Each instance consists of a design tree, an interpreted design sequence, a low level design sequence, an STL file (3D model), a `npy` file containing an array of trims, and a JSON file reporting flight statistics. Please refer to 2

Is there a label or target associated with each instance? The labels are given via the `output.json` and `trims.npy` files. These files contain labels such as whether a design has interferences and if they are able to fly. It is also possible to use the design tree or design sequence to label the 3D data. For example the low level design sequence can be used to label how many propellers are in the aircraft design or to predict which battery has been selected based on the performance file.

Is any information missing from individual instances? Everything is included. No data is missing.

Are relationships between individual instances made explicit? Individual instances are sampled independently from our generator and are therefore not explicitly related. The corpus dictionary can also be used to determine the attributes of the available components where appropriate, where this corpus is relevant to all designs.

14

Are there recommended data splits (e.g., training, development/validation, testing)? In the notebook code we provide our training, validation, and test splits but they are not explicitly recommended as the data has multiple use-cases.

Are there any errors, sources of noise, or redundancies in the dataset? The only errors come from the available fidelity of the scientific models. Please refer to [2, 30] for the limitations of the models. As a result, some of the outputs from the metadata contain NaNs. These are simple enough to filter out from the dataset where needed.

Is the dataset self-contained, or does it link to or otherwise rely on external resources? The dataset is entirely self-contained.

Does the dataset contain data that might be considered confidential? No.

Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety? Not that the authors are aware of.

Does the dataset identify any subpopulations (e.g., by age, gender)? Not applicable.

Is it possible to identify individuals (i.e., one or more natural persons), either directly or indirectly (i.e., in combination with other data) from the dataset? Not applicable.

Does the dataset contain data that might be considered sensitive in any way? No.

A.3 Collection Process

How was the data associated with each instance acquired? The data was generated by sampling our probabilistic design generator and then compiling each design into a format that was suitable for the scientific models. The scientific models provided performance metrics which we then curated into the `output.json` files and the STL files.

What mechanisms or procedures were used to collect the data? Each design took approximately 3 minutes to evaluate. As a result we estimate that it took almost 60 days of compute power to build `AircraftVerse`. As part of the pipeline, we required access to a CREO license [24].

If the dataset is a sample from a larger set, what was the sampling strategy? The dataset was sampled from our probabilistic design generator and was therefore a stochastic process.

Over what timeframe was the data collected? The data was generated using four compute nodes and therefore took over 2 weeks to generate. However, this was part of a process that took around two years of development to get to. Part of that process was building domain knowledge across the collaboration and building up the right compute architecture.

Were any ethical review processes conducted? Not applicable.

Did you collect the data from the individuals in question directly, or obtain it via third parties or other sources (e.g., websites)? Not applicable.

Were the individuals in question notified about the data collection? Not applicable.

Did the individuals in question consent to the collection and use of their data? Not applicable.

If consent was obtained, were the consenting individuals provided with a mechanism to revoke their consent in the future or for certain uses? Not applicable.

Has an analysis of the potential impact of the dataset and its use on data subjects been conducted? Not applicable.

A.4 Preprocessing/cleaning/labeling

Was any preprocessing/cleaning/labeling of the data done? For the machine learning models, we processed the design trees into a unique key-value sequence as described in Section 4. As part of evaluating the performance of each design, we recorded the performance metrics in the `output.json` file and the `trims.npy` file. We also used a combination of data-driven heuristic approaches and hard rules to remove certain designs from the final dataset. We limited designs to a maximum of 12 wings and 16 propellers. We further added a data-driven preprocessing step to remove designs that would definitely not hover. We were careful to select a threshold that did not sacrifice diversity.

Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data Each design contains the raw data as well as the processed sequences. We consider the design trees, the `STL` files and the performance files to be raw data.

Is the software that was used to preprocess/clean/label the data available? The software required to label the data requires a CREO license along with proprietary software owned by Southwest Research Institute. It is possible that the latter software will be released in the future.

A.5 Uses

Has the dataset been used for any tasks already? At the time of publication, the only use-case of our dataset can be seen in Section 4.

Is there a repository that links to any or all papers or systems that use the dataset? Not applicable at this time as this is the first instance of the data release.

What (other) tasks could the dataset be used for? The dataset could be used for anything related to modeling or understanding the behavior of aircraft. For example one could envision learning rules for good designs, inverting the design by predicting sequences from point clouds etc... . Overall the `AircraftVerse` dataset contains multiple modalities and there are many opportunities to test new models out. We take some time in the paper to highlight this.

Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses? The fidelity of the scientific models are the main factor for users of this dataset to consider when using this data set for future applications.

Are there tasks for which the dataset should not be used? Not that the authors are aware of.

A.6 Distribution

Will the dataset be distributed to third parties outside of the entity on behalf of which the dataset was created? Yes, the dataset is publicly available.

How will the dataset will be distributed? The dataset is distributed on X's webpage and on Zenodo at http://doi.org/10.5281/zenodo.6525446.

When will the dataset be distributed? The dataset was first released in 2023.

Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)? See Section B.

Have any third parties imposed IP-based or other restrictions on the data associated with the instances? No.

A.7 Maintenance

Who will be supporting/hosting/maintaining the dataset? The Neuro-Symbolic Computing and Intelligence (NuSCI) Research Group at SRI International will support and maintain the dataset.

How can the owner/curator/manager of the dataset be contacted? Please see the email addresses at the start of the paper.

Will the dataset be updated? We intend to continue to add to the dataset as we evaluate further designs.

If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so? Please contact the authors in regards to ways that the dataset could be extended. For example running different scientific models may be possible.

B Licensing description

Our dataset uses a Creative Commons Attribution-ShareAlike (CC BY-SA) license. This means that the data can be shared and/or adapted for any purpose under the condition that appropriate credit is given, and that a link to the license is provided, and that users indicate if any changes were made. For more details on the license, please refer to `https://creativecommons.org/licenses/by/4.0/`.

C Flight dynamics model

In this section, we describe the physics of the flight dynamics model (FDM). The vehicle is equipped with five major components 1) propellers, 2) motors, 3) batteries as power source, 4) wings and stabilisers, 5) body structure such as a fuselage or a board where components are connected to. The FDM considers a six-degrees of freedom where the state of the vehicle at time t is defined as follows

$$\boldsymbol{x}^t = (U, V, W, P, Q, R, \boldsymbol{q}, x, y, z, \Omega_i, Q_j). \tag{1}$$

- (U, V, W) is the velocity of the vehicle in the body frame along X, Y, and Z axis respectively.
- (P, Q, R) is the rotation of the body, i.e., representing roll, pitch, and yaw respectively.
- $\boldsymbol{q}$ is a four-dimensional vector connecting the world frame to body frame.
- (x, y, z) is the location in the world frame.
- Ω_i is the rotation for the ith motor expressed as radian/sec.
- Q_j is the power drawn from jth battery.

The flight dynamics is governed by the following equation

$$\frac{d\boldsymbol{x}}{dt} = f(\boldsymbol{x}, \boldsymbol{u}), \tag{2}$$

where $\boldsymbol{x}$ is the current state of the vehicle and $\boldsymbol{u}$ is a vector representing control signals. Each component can be controlled independently by the corresponding control signal and we assume the control signals are within range of 0 and 1. The function $f()$ represents the computation of forces. The computation of various forces as described below.

Gravity. Let us consider the velocity at the world frame is v and the body frame is V, respectively. The translation matrix connective v to V is defined as $R(\boldsymbol{q})$. Thus, $V = R(\boldsymbol{q})v$. The gravitation force is applied on the the center of mass of the vehicle and computed as

$$F_g = mgR(\boldsymbol{q})\hat{z}, \tag{3}$$

17

604   where $\hat{z} = (0, 0, 1)$ is the unit vector pointing down in the world frame.

605   **Drag.** The drag force on the vehicle's body is computed based on the vehicle's flight through the air.
606   We assume that the velocity is the vehicle's velocity in the air. Thus, the velocity is updated from the
607   body frame to account for the motion of the air being in the world frame as $U - R(\boldsymbol{q})v_w$, where v_w
608   is the wind velocity. Then the body drag is approximated as

$$F_b = -\frac{1}{2}\rho \begin{pmatrix} X_{body,uu}|U + 2v_i|(U + 2U_{in}) \\ Y_{body,vv}|V + 2V_{in}|(V + 2V_{in}) \\ Z_{body,ww}|W + 2W_{in}|(W + 2W_{in}) \end{pmatrix} \tag{4}$$

609   where $X_{body,uu}, Y_{body,vv}, Z_{body,ww}$ are the constants for each coordinate axis and for the vehicle
610   we set it equal to a constant times the presented area of the vehicle's body along the coordinate
611   axis. U_{in}, V_{in}, W_{in} are the wind velocity along X, Y, and Z directions, respectively. As for the
612   asymmetrical vehicles, center of drag may not align with the center of mass, this generates a moment
613   force given as $M = (X_{cbody} - X_{cm}) \times F_b$, where X_{cbody} is the center of body force and X_{cm} center
614   of mass.

615   **Propeller and motor modeling.** We define the geometric center of a propeller in the body frame is
616   X_p and the unit normal vector to the propeller shaft is n_p. The velocity of the air passing through the
617   stationary propeller is estimtaed as

$$V_p = (V + \vec{\Omega} \times (X_p - X_{cm})) \cdot n_p. \tag{5}$$

618   For propeller performance, we refer to pre-computed tables for each propeller providing performance
619   data with respect to spin rate and advance ratio J. The spin rate is defined by the rotations per second
620   and denoted by $n = \Omega/2\pi$. The advanced ratio is computed as $J = V_p/(nD)$. where D is the
621   diameter of the propeller. The thrust and power of the propeller are estimated as

$$F_p = C_T(n, J)\rho n^2 D^4, P = C_p(n, J)\rho n^3 D^5, \tag{6}$$

622   where F_p and P are thrust and power. $C_T()$ and $C_p()$ are pre-computed performance values depending
623   on n and J. The total force from the propeller is computed by summing over all propellers.

624   The torque from the motor is estimated as $\tau_{motor} = K_T(I - I_0)$, where I is the current, I_0 is the
625   idle current, and K_T is the torque constant. The relation between the current and voltage is given by

$$V_{motor} = V_{battery}u_c = \frac{\Omega}{K_V} + IR_w \tag{7}$$

626   where V_{motor} is the voltage applied to the motor, $V_{battery}$ is the battery voltage, and $u_c = [0, 1]$ is
627   the control signal driving the motor. K_V is the RPM constant and R_w is the resistance.

628   Note that the flight dynamics simulation model is currently under construction, and there are some
629   assumptions that would affect the flight characteristics and performance of a given design outside of
630   this particular FDM. For example, the parametric connectors are currently made of 3d ABS plastic
631   which accurately reflects the mass properties of such a material, but may not hold up to the structural
632   loads experienced during flight, which are not currently modeled. Further development will continue
633   to address these simulation gaps.

D   Additional experiment details

635   For the baseline experiments listed in this paper, we randomly shuffle the entire dataset and then split
636   the data into train, validation, and test splits of 70 %, 10 %, and 20 % respectively. For the regression
637   tasks, we normalize the data using the mean and standard deviation of the data. Where necessary we
638   remove anomalous data points. For the mass prediction, we remove designs with a mass greater than
639   35 Kg from the data set. When we report the results in 1 for MSE, we use the unnormalized values.

D.1   LSTM architecture

641   The LSTM baseline model consists of a linear input layer followed by an LSTM layer (`nn.LSTM` in
642   PyTorch) with an input size of 512, a hidden size of 512. The LSTM layer is then connected to a

final linear layer, via a dropout layer with a value of 0.5, that takes in all the outputs of the LSTM layer and provides the one-dimensional prediction. During training we use the PyTorch in-built SGD optimizer with a learning rate of 0.01 and set the number of epochs to 2000, where we save the model with the best validation loss.

D.2 Transformer encoder architecture

The transformer encoder consists of a linear encoding layer followed by a positional encoding layer. The output of the positional encoding is then passed through a transformer encoding layer (`nn.TransformerEncoder` in PyTorch) with 8 layers and 2 heads. The final token from the transformer encoder is then passed through a linear layer that goes from the embedding dimension of 200 to a single output dimension. During training we use the PyTorch in-built SGD optimizer with a learning rate of 0.1 and set the number of epochs to 1000, where we save the model with the best validation loss.

D.3 Graph convolutional neural network

Apart from the aforementioned models on sequence data, we also consider a model that directly operates on the point cloud. We represent the point clouds as a set of 3D points and apply a graph convolutional network (GCN) [17, 31] to make predictions on them. We build the graphs with the points as nodes where nodes are represented by their 3D coordinate. Each node is connected to K nearest neighbors (K=40). Note that the GCN model captures the structural information from the point clouds and does not have access to the specification of the components. Our GCN has four convolutional layers followed by a fully connected layer. We train the GCN with Adam optimizer for 200 epochs.

E Automated data generation

As part of automating our data generation pipeline for `AircraftVerse`. We defined a design grammar. Building a grammar to define a CPS requires significant domain expertise and `AircraftVerse` was no exception. Our focus was to build a procedural generator of designs that produced topologically valid aircraft designs. I.e. designs that met the requirements needed to be evaluated by the scientific models. Appendix G shows an example of a valid design tree. Once we built a valid procedural generator (with uniform distributions over all choices), we added the following heuristics: limit designs to having a maximum of 12 wings; limit designs to having a maximum of 16 propellers; set the probability of a recursive structure to be 0.25, such that we do not get infinitely long recursive structures. We then generated half our dataset using our procedural generator. Thereafter, we trained a transformer-based classifier to identify designs from their sequences that were extremely unlikely to hover (similar to the benchmark outlined in Section D). The next stage of generation then used this classifier as a weak filter (calibrated according to a recall of 90 %) to increase the number of hovering vehicles in the dataset. Components of this process are highlighted in our previous technical report [6]. Overall this process is an accumulation of over two years of work in building the right domain knowledge and compute infrastructure to ensure that the data remains diverse, interesting, and useful.

F Aircraft grammar and components list

In this section, we include the list of options available in the data set for our grammar that is included in the `design_seq.json` and `design_tree.json` file. Each token in the sequence conists of a key-value pair as shown in Figure 2. The available keys are as follows:

19

Table 2: Key-value descriptions.

KEY	DESCRIPTION	VALUE OPTIONS
node_type	Main structural components of aircraft, like the central hub and connector arms.	'AngledPropArm', 'AngledWingArm', 'BendSegment', 'BranchSegment_Asym', 'BranchWithTopSegment_Asym', 'ConnectedHub2_Sym_Wide', 'ConnectedHub3_2_1', 'ConnectedHub4_2_2', 'ConnectedHub4_Sym', 'ConnectedHub6_1_2_2_1', 'ConnectedHub6_2_2_2', 'ConnectedHub6_Sym', 'DoubleBendSegment', 'PropArm', 'SidewaysBendSegment', 'SidewaysBendWithTopSegment', 'WingArm'.
propType	Propeller type.	'apc_propellers_10x5', 'apc_propellers_10x6', 'apc_propellers_10x7', 'apc_propellers_11x5_5', 'apc_propellers_11x8', 'apc_propellers_12x12', 'apc_propellers_12x6', 'apc_propellers_12x8', 'apc_propellers_13x10', 'apc_propellers_13x4', 'apc_propellers_13x5_5', 'apc_propellers_13x6_5', 'apc_propellers_13x8', 'apc_propellers_14x7', 'apc_propellers_14x8_5', 'apc_propellers_15x10', 'apc_propellers_15x4', 'apc_propellers_16x10', 'apc_propellers_16x4', 'apc_propellers_18x10', 'apc_propellers_18x8', 'apc_propellers_19x10', 'apc_propellers_20x10', 'apc_propellers_4_1x4_1', 'apc_propellers_4_75x4_75', 'apc_propellers_5_5x4_5', 'apc_propellers_5x3', 'apc_propellers_5x5', 'apc_propellers_5x7_5', 'apc_propellers_6x4', 'apc_propellers_6x6', 'apc_propellers_7x4', 'apc_propellers_7x5', 'apc_propellers_7x6', 'apc_propellers_8x6', 'apc_propellers_8x8', 'apc_propellers_9x4_5', 'apc_propellers_9x6'
motorType	Motor type.	'kde_direct_KDE2306XF2550', 'kde_direct_KDE2315XF885', 'kde_direct_KDE2315XF965', 'kde_direct_KDE2814XF_515', 'kde_direct_KDE2814XF_775', 'kde_direct_KDE3510XF_475', 'kde_direct_KDE3510XF_715', 'kde_direct_KDE3520XF_400', 'kde_direct_KDE4012XF_400', 'kde_direct_KDE4014XF_380', 'kde_direct_KDE4213XF_360', 't_motor_AS2308KV1450', 't_motor_AS2308KV2600', 't_motor_AS2312KV1150', 't_motor_AS2312KV1400', 't_motor_AS2317KV1250', 't_motor_AS2317KV1400', 't_motor_AS2317KV880', 't_motor_AS2814KV1050', 't_motor_AS2814KV1200', 't_motor_AS2814KV2000', 't_motor_AS2814KV900', 't_motor_AS2820KV1050', 't_motor_AS2820KV1250', 't_motor_AS2820KV880', 't_motor_AT2308KV2600', 't_motor_AT2310KV2200', 't_motor_AT2312KV1150', 't_motor_AT2312KV1400', 't_motor_AT2317KV1250', 't_motor_AT2317KV1400', 't_motor_AT2317KV880', 't_motor_AT2321KV1250', 't_motor_AT2321KV950', 't_motor_AT2814KV1050', 't_motor_AT2814KV1200', 't_motor_AT2814KV900', 't_motor_AT2820KV1050', 't_motor_AT2820KV1250', 't_motor_AT2820KV880', 't_motor_AT2826KV900', 't_motor_AT3520KV550', 't_motor_AT3520KV720', 't_motor_AT3520KV850', 't_motor_AT3530KV580', 't_motor_AT4120KV500', 't_motor_AT4120KV560', 't_motor_AT4125KV540', 't_motor_AT4130KV450', 't_motor_MN2212KV780', 't_motor_MN2212KV920', 't_motor_MN3110KV470', 't_motor_MN3110KV700', 't_motor_MN3110KV780', 't_motor_MN3508KV380', 't_motor_MN3508KV580', 't_motor_MN3508KV700', 't_motor_MN3510KV360', 't_motor_MN3510KV630', 't_motor_MN3510KV700', 't_motor_MN3515KV400', 't_motor_MN3520KV400', 't_motor_MN4010KV370', 't_motor_MN4010KV475', 't_motor_MN4010KV580', 't_motor_MN4012KV340', 't_motor_MN4012KV400', 't_motor_MN4012KV480', 't_motor_MN4014KV330', 't_motor_MN4014KV400', 't_motor_MN5208KV340', 't_motor_MN5212KV340', 't_motor_MN5212KV420', 't_motor_MT22081100KV', 't_motor_MT2216V2800KV'
batteryType	Battery type.	'TurnigyGraphene1000mAh2S75C', 'TurnigyGraphene1000mAh3S75C', 'TurnigyGraphene1000mAh4S75C', 'TurnigyGraphene1000mAh6S75C', 'TurnigyGraphene1200mAh6S75C', 'TurnigyGraphene1300mAh3S75C', 'TurnigyGraphene1300mAh4S75C', 'TurnigyGraphene1400mAh3S75C', 'TurnigyGraphene1400mAh4S75C', 'TurnigyGraphene1500mAh3S75C', 'TurnigyGraphene1500mAh4S75C', 'TurnigyGraphene1600mAh4S75C', 'TurnigyGraphene1600mAh4S75CSquare', 'TurnigyGraphene2200mAh3S75C', 'TurnigyGraphene2200mAh4S75C', 'TurnigyGraphene3000mAh3S75C', 'TurnigyGraphene3000mAh6S75C', 'TurnigyGraphene3000mAh6S75C', 'TurnigyGraphene4000mAh3S75C', 'TurnigyGraphene4000mAh4S75C', 'TurnigyGraphene4000mAh6S75C', 'TurnigyGraphene5000mAh3S75C', 'TurnigyGraphene5000mAh6S75C', 'TurnigyGraphene5000mAh6S75C', 'TurnigyGraphene6000mAh3S75C', 'TurnigyGraphene6000mAh4S75C', 'TurnigyGraphene6000mAh6S75C'
wingType	Wing type.	'NACA_0012', 'NACA_0015', 'NACA_0018', 'NACA_0021', 'NACA_0025', 'NACA_2212', 'NACA_2306', 'NACA_2309', 'NACA_2312', 'NACA_2315', 'NACA_2406', 'NACA_2409', 'NACA_2412', 'NACA_2415', 'NACA_2418', 'NACA_2421', 'NACA_2506', 'NACA_2509', 'NACA_2512', 'NACA_2515', 'NACA_2518', 'NACA_2521', 'NACA_2612', 'NACA_2712', 'NACA_4212', 'NACA_4306', 'NACA_4309', 'NACA_4312', 'NACA_4315', 'NACA_4318', 'NACA_4321', 'NACA_4406', 'NACA_4409', 'NACA_4412', 'NACA_4415', 'NACA_4418', 'NACA_4421', 'NACA_4506', 'NACA_4509', 'NACA_4512', 'NACA_4515', 'NACA_4518', 'NACA_4521', 'NACA_4612', 'NACA_4712', 'NACA_6212', 'NACA_6306', 'NACA_6309', 'NACA_6312', 'NACA_6315', 'NACA_6318', 'NACA_6321', 'NACA_6406', 'NACA_6409', 'NACA_6412', 'NACA_6415', 'NACA_6418', 'NACA_6421', 'NACA_6506', 'NACA_6509', 'NACA_6512', 'NACA_6515', 'NACA_6518', 'NACA_6521', 'NACA_6612', 'NACA_6712'
servoType	Servo type for control of flaps and ailerons.	'Hitec_D485HW', 'Hitec_D89MW', 'Hitec_D954SW', 'Hitec_HS_225BB', 'Hitec_HS_225MG', 'Hitec_HS_40', 'Hitec_HS_45HB', 'Hitec_HS_5055MG', 'Hitec_HS_5065MG', 'Hitec_HS_5070MH', 'Hitec_HS_5087MH', 'Hitec_HS_5245MG', 'Hitec_HS_53', 'Hitec_HS_5496MH', 'Hitec_HS_55', 'Hitec_HS_5565MH', 'Hitec_HS_625MG', 'Hitec_HS_645MG', 'Hitec_HS_65HB', 'Hitec_HS_65MG', 'Hitec_HS_70MG', 'Hitec_HS_7235MH', 'Hitec_HS_7245MH', 'Hitec_HS_81', 'Hitec_HS_82MG', 'Hitec_HS_85BB', 'Hitec_HS_85MG'

Continued on next page

Table 2 – continued from previous page

KEY	DESCRIPTION	VALUE OPTIONS
wingRot	The wing's angle.	Float: [0, 360]
chord	The wing's chord.	Float: [100, 500]
span	The wing's span.	Float: [200, 2000]
armLength	A context specific arm length generally associated with node_types such as 'PropArm' and 'WingArm'.	Float: [50, 500]
arm1Length	A more specific arm length associated with arms that include a bend.	Float: [50, 500]
arm2Length	A more specific arm length associated with arms that include a bend.	Float: [50, 200]
angle	A context specific angle that corresponds to angles in the vehicle construction.	Float: [-135, 360]
offset	A context specific offset value that can correspond to structural components such as the wing offset and the flange offset.	Float: [-200, 90]
x1_offset	Offset in the x-direction of Battery on the central plate.	Float: [0, 120]
z1_offset	Offset in the z-direction of Battery on the central plate.	Float: [0, 120]
tube_offset	The wing tube offset.	Float: [10, 90]

684

G Aircraft design summary

In the pages following, we include the complete representation of an individual design when accessed from AircraftVerse where we print the:

- design_tree.json
- design_seq.json
- cadfile.stl
- pointCloud.npy
- output.json
- trims.npy

In addition, we show how to access the design corpus and provide an example of printing the specification of a battery.

DataDemo-Paper

May 26, 2023

```python
[1]: import os
     import json
     import pickle
     import numpy as np
     import sys
     sys.path.append("./code")
     import util
     import matplotlib.pyplot as plt
     %matplotlib inline


     path = '<dataset path>'
     corpus = '<Corpus Path>'
     design = 'design_n'
```

Show data folder structure

```python
[2]: data_files = os.listdir(os.path.join(path,design))
     print(data_files)
```

```
['design_seq.json', 'design_tree.json', 'design_low_level.json', 'Geom.stp',
'cadfile.stl', 'trims.npy', 'output.json', 'pointCloud.npy']
```

Show design tree

```python
[3]: tree = open(os.path.join(os.path.join(path,design), 'design_tree.json'))
     tree_json = json.load(tree)
     tree_json_print = json.dumps(tree_json, indent=4)
     tree.close()
     print(tree_json_print)
```

```
{
    "generator_version": "UAV2_gen12",
    "name": "design_12028_e437c0c4835841a99d48808aa8734bc7_s",
    "hub": {
        "node_type": "ConnectedHub4_Sym",
        "mainSegment": {
            "node_type": "PropArm",
            "armLength": 413.5996130739701,
            "motor": {
```

```
697                    "motorType": "t_motor_AT3530KV580"
               },
               "prop": {
                   "propType": "apc_propellers_18x8"
               },
               "flange": {
                   "offset": 41.65045980035022,
                   "angle": 0.0
               }
           }
       },
       "fuselageWithComponents": {
           "node_type": "SingleBatteryFuselageWithComponents",
           "battery": {
               "batteryType": "TurnigyGraphene2200mAh3S75C"
           },
           "fuselage": {
               "length": 106.0,
               "vertDiameter": 52.5,
               "horzDiameter": 332.4,
               "floorHeight": 8.75,
               "batteryX": 0.0,
               "batteryY": 6.0,
               "rpmX": 0.0,
               "rpmY": -100.8,
               "autoPilotX": 0.0,
               "autoPilotY": 44.5,
               "currentX": 0.0,
               "currentY": -80.65,
               "voltageX": 0.0,
               "voltageY": 98.3,
               "gpsX": 0.0,
               "gpsY": -40.5,
               "varioX": 0.0,
               "varioY": 76.65
           }
       }
   }
```

Design Corpus

- One can use the design corpus to add context to the component types. For example, here is a print out of the attributes of a single battery.

```
[4]: corpus_file = open(corpus, "rb")
     corpus_dic = pickle.load(corpus_file)
     corpus_file.close()
     print(corpus_dic.keys())
```

```
print('\nBattery Example:\n')
print('Number of batteries: ', len(corpus_dic['Battery'].keys()))
print('Battery example: TurnigyGraphene2200mAh3S75C:\n')
for key, value in corpus_dic['Battery']['TurnigyGraphene2200mAh3S75C'].items():
    print('{:20s}: {:}'.format(key, value))
```

```
dict_keys(['Servo', 'GPS', 'ESC', 'Wing', 'Sensor', 'Propeller', 'Receiver',
'Motor', 'Battery', 'Autopilot'])


Battery Example:

Number of batteries:  36
Battery example: TurnigyGraphene2200mAh3S75C:

CAPACITY             : 2200.0
CONT_DISCHARGE_RATE  : 75.0
COST                 : 32.73
LENGTH               : 106.0
PEAK_DISCHARGE_RATE  : 150.0
THICKNESS            : 31.0
VOLTAGE              : 11.1
WEIGHT               : 0.23
WIDTH                : 35.0
```

Design Sequence

- As described in the paper, we convert the design tree into a design sequence which we find to be useful for sequence based machine learning approaches.
- Here we show an example sequence that corresponds to the tree above.

```
[5]: seq = open(os.path.join(os.path.join(path,design), 'design_seq.json'))
     seq_json = json.load(seq)
     seq_json_print = json.dumps(seq_json, indent=4)
     seq.close()
     print(seq_json_print)
```

```
[
    {
        "generator_version": "UAV2_gen12"
    },
    {
        "name": "design_12028_e437c0c4835841a99d48808aa8734bc7_s"
    },
    {
        "node_type": "ConnectedHub4_Sym"
    },
    {
        "node_type": "PropArm"
```

},
{
    "armLength": 413.5996130739701
},
{
    "motorType": "t_motor_AT3530KV580"
},
{
    "propType": "apc_propellers_18x8"
},
{
    "offset": 41.65045980035022
},
{
    "angle": 0.0
},
{
    "node_type": "SingleBatteryFuselageWithComponents"
},
{
    "batteryType": "TurnigyGraphene2200mAh3S75C"
},
{
    "length": 106.0
},
{
    "vertDiameter": 52.5
},
{
    "horzDiameter": 332.4
},
{
    "floorHeight": 8.75
},
{
    "batteryX": 0.0
},
{
    "batteryY": 6.0
},
{
    "rpmX": 0.0
},
{
    "rpmY": -100.8
},
{
    "autoPilotX": 0.0

4

```
700,
    {
        "autoPilotY": 44.5
    },
    {
        "currentX": 0.0
    },
    {
        "currentY": -80.65
    },
    {
        "voltageX": 0.0
    },
    {
        "voltageY": 98.3
    },
    {
        "gpsX": 0.0
    },
    {
        "gpsY": -40.5
    },
    {
        "varioX": 0.0
    },
    {
        "varioY": 76.65
    }
]
```
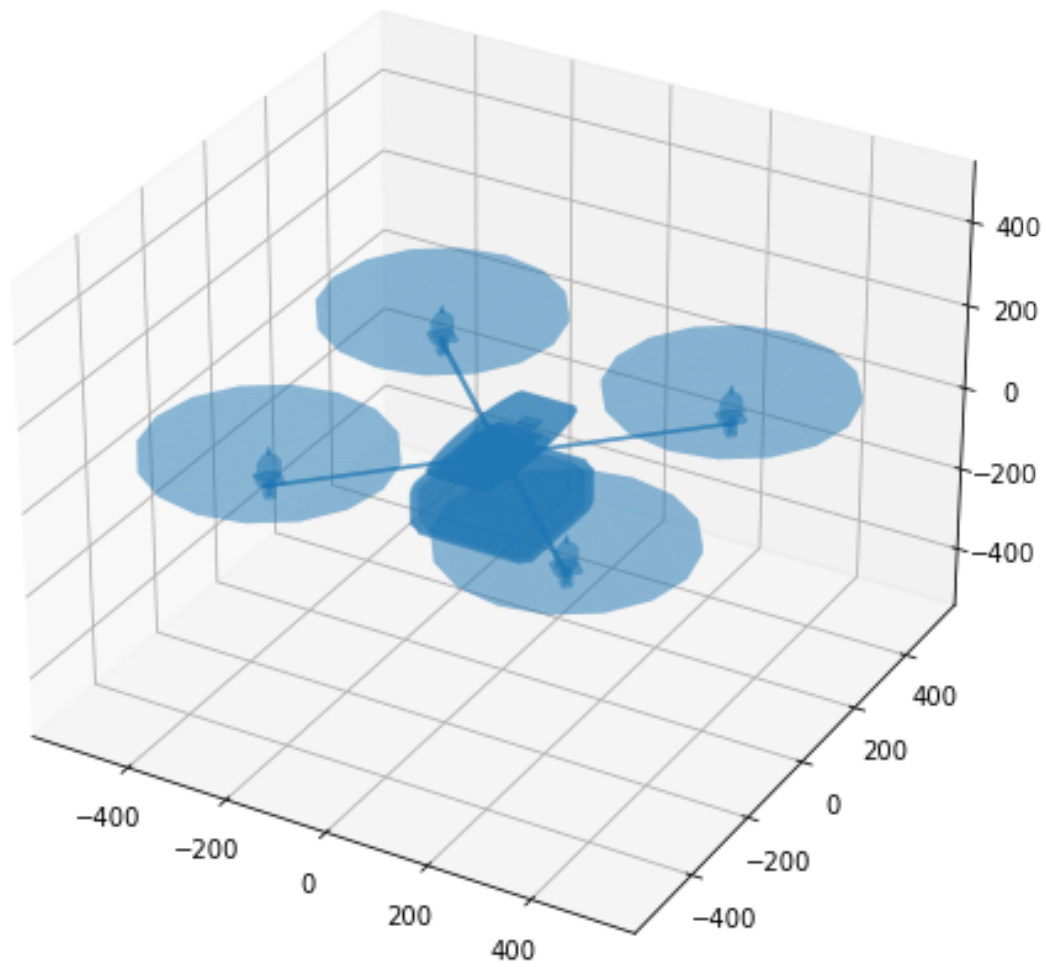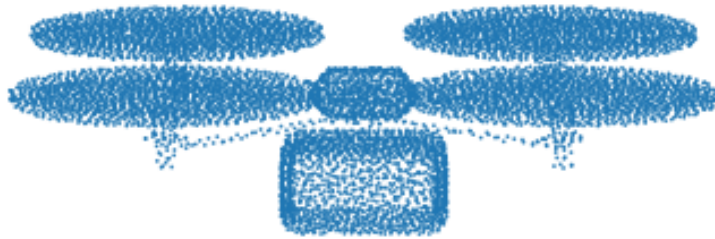
0.0.1 STL plot

```
[6]: util.plot_stl(path, design)
```

701

0.0.2 Point Cloud

- One can easily convert the STL file into a point cloud... Here is one we made earlier and have included in the data set.

```
[7]: util.plot_pointCloud(path, design)
```

0.0.3 Output File

```
[8]: seq = open(os.path.join(os.path.join(path,design), 'output.json'))
     seq_json = json.load(seq)
     seq_json_print = json.dumps(seq_json, indent=4)
     seq.close()
     print(seq_json_print)
```

```
{
    "Interferences": 2,
    "Mass": 3.4330805367769828,
    "Batt_amps_ratio_MFD": 0.72659409,
    "Batt_amps_ratio_MxSpd": 0.906760275,
```

```
703 "Distance_MxSpd": 1228.10852,
    "Max_Distance": 1268.38354,
    "Hover_Time": 83.8901367,
    "Max_Speed": 29.0,
    "Max_uc_at_MFD": 0.738918006,
    "Mot_amps_ratio_MFD": 0.602965593,
    "Mot_amps_ratio_MxSpd": 0.807905734,
    "Mot_power_ratio_MFD": 0.310742617,
    "Mot_power_ratio_MxSpd": 0.416359991,
    "Power_MFD": 872.750122,
    "Power_MxSpd": 1350.86023,
    "Speed_MFD": 24.0
}
```

0.0.4   Plot the trims for the UAV design

```
[9]: trims = np.load(os.path.join(os.path.join(path,design),'trims.npy'))
     names = ['distance',
      'flight_time',
      'pitch_angle',
      'max_uc',
      'thrust',
      'lift',
      'drag',
      'current',
      'total_power',
      'frac_amp',
      'frac_pow',
      'frac_current']
     plt = util.plot_trim_stats(trims, names, fs = 19, figsize = (23,2.))
     plt.show()
```