Customizable Turbofan Engine Component in OpenVSP

Natalie G. Bretton[1]
University of Virginia

Open Vehicle Sketch Pad, or OpenVSP, is a versatile open-source parametric tool for aircraft design. Originally developed by NASA, OpenVSP is a crucial part of modeling aircraft and designing prototypes that are later exported into computational fluid dynamics (CFD) software. Although it excels at importing various aircraft components, the lack of a dedicated engine component makes it difficult to ensure accurate CFD calculations, especially when it comes to engine-airframe integration and drag prediction. This paper details the design of a customizable turbofan engine component for OpenVSP, allowing users to incorporate engine design into their aircraft model alongside all the other features. The component has fully customizable parameters and special features that ensure engine effects can be incorporated into design and not overlooked. This model is currently available on GitHub[2], and is easily incorporated into OpenVSP, improving the accuracy of aircraft modeling.

I. Introduction

When designing aircraft, the process always starts with the most fundamental concepts that govern flight. Design often begins with the analysis of parameters like surface area and how they affect the drag an aircraft will experience. These calculations can easily be approximated by analyzing the basic geometric shapes that make up said design—flat planes, cylinders, etc. Due to information that is readily available now, the amount of drag an object of known size would experience in different situations can be calculated. By analyzing the combined surface area of the wings, fuselage, and other entities of a plane, a drag estimate can be mathematically approximated. These kinds of basic ideas have been continuously developing and improving, and the technology available now allows for very detailed analysis of aircraft aerodynamics. The problem arises when the accuracy of the calculations vastly exceeds the accuracy of the geometry itself. Extremely detailed and advanced calculations of drag and other aerodynamics will only be useful if the model on which these calculations are being run is accurate and realistic.

When it comes to engine modeling, important details are often overlooked in comparison to the rest of the aircraft, and simple, cylindrical engine components are added to more detailed aircraft models. With the currently available features, the usual technique is to either model the engine as a generic hollow tube with jump conditions assuming a uniform flow stream, or to skip the engine in its entirety. This is troublesome because the analysis that results from these models will never reach its maximum potential when it comes to accuracy. The design of the customizable turbofan engine component for OpenVSP aims to help resolve this lack of accuracy. This component, like other custom components in OpenVSP, allows the user to add the engine to a model with one click and then modify it using different parametric sliders. Although there are options to add wings, fuselages, and many other features to models, there is unfortunately no engine component. Currently, an engine can be designed using the stack feature; however, this involves the user going through and defining the figure cross-section by cross-section, which is a very time-consuming process. Additionally, the technique of defining an engine with stacks malfunctions if a parameter is adjusted in a way that is not ideal for the software. Consequently, there is a need for a model that can easily be added to the model like any other OpenVSP component, can be modified intuitively and visually, and is more automated than something like a stack.

---

[1]Student, University of Virginia, Mechanical Engineering
[2]Link to code: https://github.com/NatBretton/CustomizableTurbofanEngineComponent_OpenVSP/tree/main

Although this custom component is not meant to be a one hundred percent precise engine model, by allowing the user to quickly define things like inlet strength, maximum diameter, and nacelle length, a more accurate geometry can be efficiently created. The need for a customizable engine component is only growing as people are increasingly using computational fluid dynamics (CFD). Models are being imported; however, the accuracy of their geometry is not catching up to the fidelity of the solver. In other words, very good calculations are done on subpar or unrealistic models. The goal of this engine component is to bridge the gap between user-friendly, visual, engine design and accurate aircraft modeling and CFD calculations using OpenVSP.

A. OpenVSP

OpenVSP, or Open Vehicle Sketch Pad, is a parametric aircraft geometry tool that allows users to quickly draft aircraft like the one shown below (Figure 1). This is one of the main aircraft design programs used by NASA and is effective at allowing users to construct the framework of planes efficiently. Using this software, the user can import various custom components and modify important parameters in a simple, visual, and intuitive way [1].

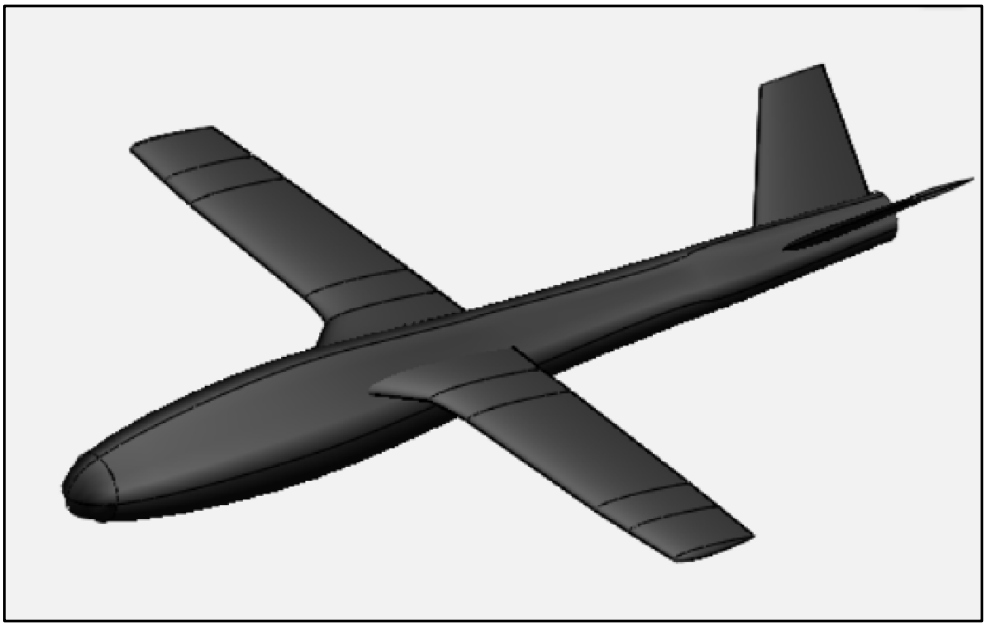


Figure 1. Basic Demonstration of OpenVSP Aircraft Modeling. *An example of a plane modeled in OpenVSP is shown above. Some of the different custom component features that are available, like the wings and fuselage, can be observed.*

There were several ways that the engine design could have been approached, including built-in components and advanced linking, but it was decided that using a custom VSP part would be the easiest and most effective way. Built-in components in VSP are more difficult to design, and modifying them becomes a bit more complicated, limiting the user's freedom. Additionally, advanced linking was experimented with, however, it did not allow for enough control of the different parameters. It was concluded that designing a custom part was the best way to give the user optimal control over the engine while ensuring that the coding process is manageable. It is also a very straightforward process to alter, customize, or personalize a custom VSP part, so if a certain project calls for a feature of the engine to be modified, the user can easily do that.

II. Methodology

A. Advanced Linking

As mentioned before, a process called advanced linking was experimented with in the search for an appropriate engine design method. Advanced linking is a capability of OpenVSP that allows a user to relate two or more

components to each other through equations. This makes it so that if one feature of a part is altered, a separate feature will vary with it according to the imputed equation. For example, Figure 2 shows a cylinder where diameter and length were linked together such that the cross-sectional area of the cylinder was always three times its length. Therefore, when the inputs (diameters) were changed, the length changed with it.

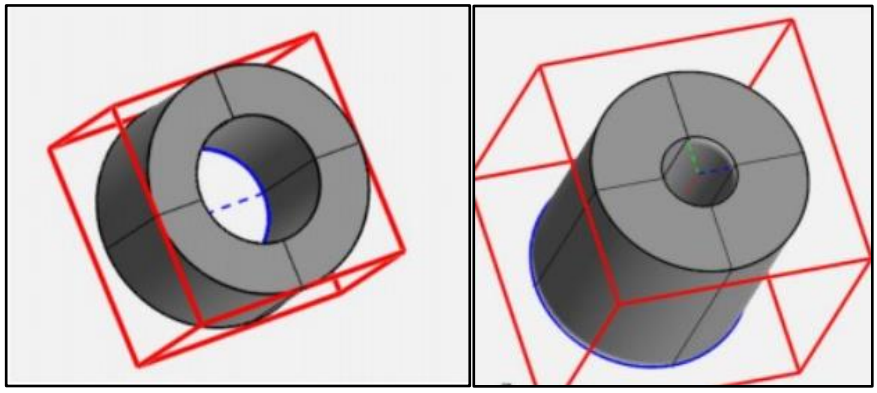


Figure 2. Advanced linking of two cylinders. *Figure 2 demonstrates how features of different components can be linked together. By adjusting the diameter of the cylinder on the left, the length and diameter of the cylinder on the right is adjusted accordingly.*

Theoretically, this linking could correlate different parameters with each other in a multitude of different ways, only limited by the difficulty of the mathematical equations used. Figure 3 shows how the advanced linking for the cylinder example above is set up, with a section for input parameters, output parameters, and the code relating the two. Potentially, this kind of linking would have been used for something like inlet design or nacelle shaping, where there is logically a certain shaping that works best for aerodynamics. If a user wanted to adjust a parameter, say inlet length, then the thickness and shaping of the cowl would automatically be changed in a way that kept the inlet a proper shape for airflow.

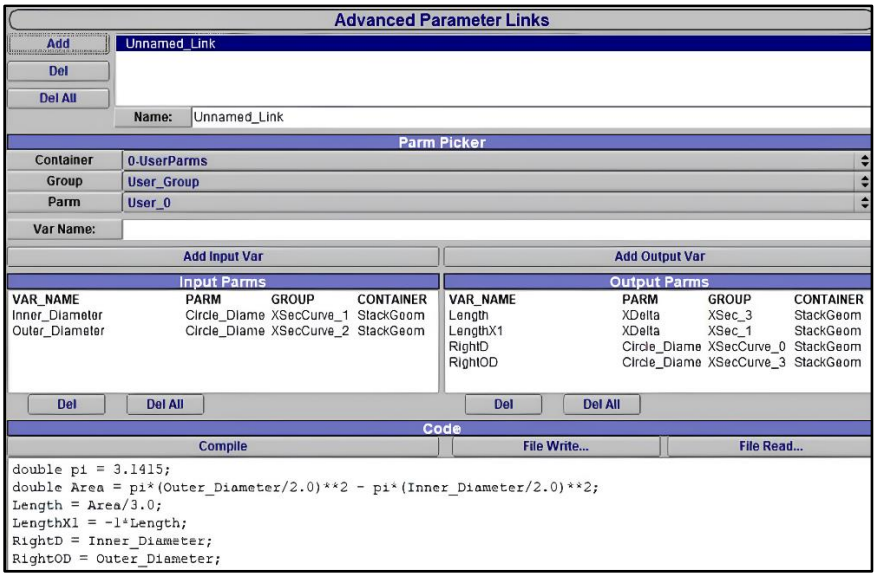


Figure 3. Graphical User Interface used for advanced linking. *The figure above shows how different parameters can be linked by adding defining equations to create a relationship between them. Inputs can be seen in the top left, and outputs in the top right.*

In the end, advanced linking was not used because one of the main goals was to make a fully customizable engine. If the parts were all linked to each other given specific equations already set in place, it took away a certain aspect of freedom that should be available. Additionally, the math and equations necessary to link a whole engine model would become complicated and hard to keep track of quickly. It was decided that advanced linking is better used for small-scale parameter linking instead of a whole turbofan engine model. However, this feature is still a very useful capability of OpenVSP and can be applied individually by the user to the customizable engine VSP part.

B. Custom Component Setup

After exploring some of the other options previously mentioned, it was decided that the custom VSP component was the most all-encompassing way to design a part with all the necessary features. The custom component feature allows users to seamlessly add their geometries to the already available ones in OpenVSP. These parts are coded using an open-source language called AngelScript that follows a C++ syntax. The exact application program interface (API) is available online and accessible for anyone to look at and use. This makes it easy for anyone to design a custom component specifically for what they are working on. Since these custom components are made from scratch, the turbofan engine component was designed such that every part of it can be adjusted but the user, with more attention added to the most integral parts of the engine.

The custom component was by establishing each of the cross-sections on the engine and using variables to define the adjustable parameters. These variables were then organized into the graphical user interface (GUI) so that the dimensions and locations of the cross-sections could be adjusted using sliders. This provides a straightforward and visual way for users to modify their engine design while ensuring that all of the complicated code remains behind the scenes. This code can be done in any text editor as long as it is saved as a .vsppart file to the "CustomScripts" folder of the OpenVSP application. This folder contains all of the scripts for the currently available parts in VSP like the fuselage, wings, and stacks. When the user runs OpenVSP it calls all of the custom scripts in the folder and they will automatically be available to add to the model. Figure 4 shows the appearance of the custom scripts folder and the GUI in OpenVSP with all of the components.

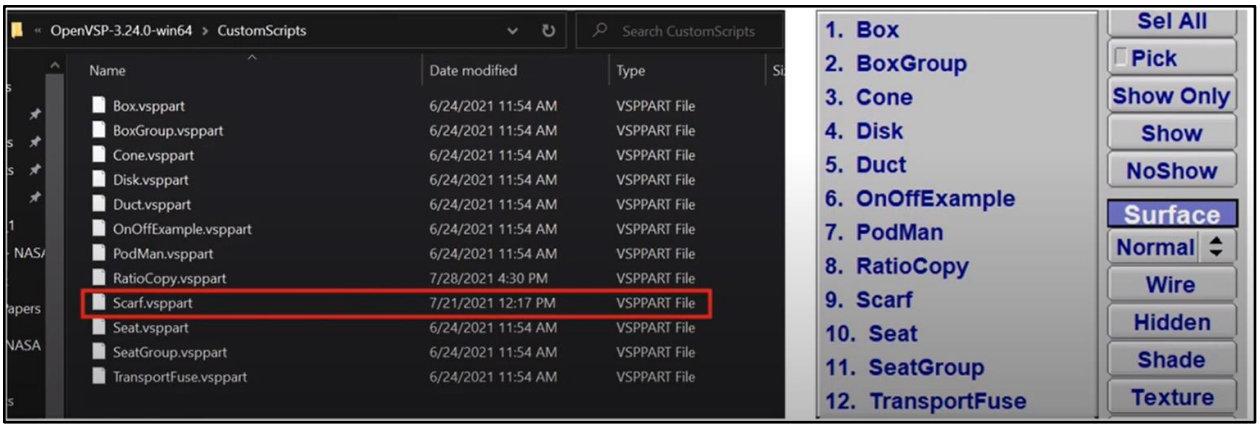


Figure 4. CustomScripts setup as seen on a computer. *Figure 4 shows how the CustomScript for the engine component can be saved to the OpenVSP application folder on a computer and then opened in OpenVSP as a custom component. The program 'Scarf' represents a model of the engine with scarfing, and when saved to the CustomScripts folder of the OpenVSP application, the app can be opened and the scarf component can be imported with one click.*

C. Engine Component Layout

The Custom Component was designed with three main tabs that cater to an intuitive modeling process for the user. There are a few tabs that are included in all OpenVSP custom parts that involve placement of the part, rotation, color, cross-section tessellation, etc. However, the three main tabs unique to the custom engine component are the Engine, Nacelle, and Strengths tabs. The goal of this setup was to make it as easy and swift as possible for a user to customize an engine specific to their needs. It is recommended that the user start the modeling process in the "Engine" tab. This is where the main geometry of the engine will be defined. If a user wants to design an exact replica of an engine on

the market they have two options. One is a special feature that OpenVSP has allowing the user to take a technical image and add it to the background as a guide while creating the engine model. The second option would be to look up the technical specifications of the engine and use those values to define the engine model. For the engine tab, these specifications would include things like the size of the spinner, the diameter of the fan face, and generic features of the internal engine components.

Next, the user would move on to define the features included in the Nacelle tab. Although the engine tab was made to focus on the interior and mechanical portions of the engine, the nacelle tab focuses on the external geometry of the engine's shell. This tab includes parameters like inlet defining features, nacelle shaping, and exhaust areas. The detailed definition of these exhaust areas allows the user to simulate the engine's boundary conditions more accurately and therefore provide more beneficial and realistic calculations. These options are one of the main goals of this model and emphasize the difference between the customizable engine component and just using a simple stack or cylinder with jump conditions to replace an engine model. Shaping in the nacelle tab is what allows the engine to be truly modeled as a dual-flow engine instead of a simple flow-through model.

The last tab is the strengths tab and focuses on the more detailed geometry of the engine when it comes to shaping curvature. The term strength is an OpenVSP concept that is not intuitive and can be difficult to visualize when modeling. It refers to the curvature of a line by numerically defining how far out a line tangentially goes from its cross-section before curving to meet the next cross-section. If a user were to design an engine from scratch and try to control the strengths of each individual cross-section this process can quickly get confusing and overwhelming. By adding these strength components into a model in a well-defined tab with an intuitive GUI, users can tune the fine details of the model more visually.

III. Engine Component Features

The most difficult part of the engine design process was determining what parameters should be available to modify and which should be defined behind the scenes. During this process, decisions had to be made about how and what different variables would be controlled. For example, choosing to set a bypass ratio as an adjustable area ratio, or letting the user control the inner and outer diameters separately. These decisions were made to appeal to a target audience that understands aerospace terminology like nacelle length and bypass ratio but might not be familiar enough with OpenVSP to know the specific title that a given cross-section might have. By making the engine component automated, and choosing the most integral components to automate with visual, intuitive sliders, users with experience in many different realms of aerospace design can easily catch on and quickly develop an engine model regardless of previous experience with OpenVSP.

Given these difficult decisions about how to define certain variables, the engine design process might not perfectly fit every case scenario. However, since the code is so easy to modify, it is a simple process to go and alter how certain parameters are controlled. Additionally, users can add more specificity and tie several sliders together using the advanced linking techniques that were previously mentioned. All of this makes the model incredibly adaptable for all different kinds of users and their unique projects. The customizable features of the engine, nacelle, and strengths tab are introduced below.

A. Engine Tab

The engine tab controls a majority of the engine's internal geometry. The goal here was to work from the inside out, where things like bypass ratio and turbine diameter are defined first. If the user follows the tabs in order, the design process will go smoothly, and they should not have to readjust parameters due to changes made by altering a different parameter. Values for the engine tab might be taken from the engineering specifications of a given model. In that case, if a user knows the exact size of an engine component, they can directly type the value in to the right of the slider. Figure 5 shows the Engine tab and includes only a few parameters to define the basic size of the interior geometry. It includes the length and diameter of the spinner at the front of the engine, the diameter of the fan face, and the engine length and turbine diameter at the exit. Note that the spinner diameter and the fan diameter are the two cross-sections that define the area of the fan face, and thus define the engine bypass ratio. The decision was made to keep these values defined by diameter and not area to give the user more control over the engine. However, if a user prefers to look at the parameters as an area value, all they have to do is enter a quick equation into the advanced linking settings, and they can tie the two features together. Alternatively, a minor modification could be made to the code itself to add a slider specifically for bypass ratio in place of the diameters.

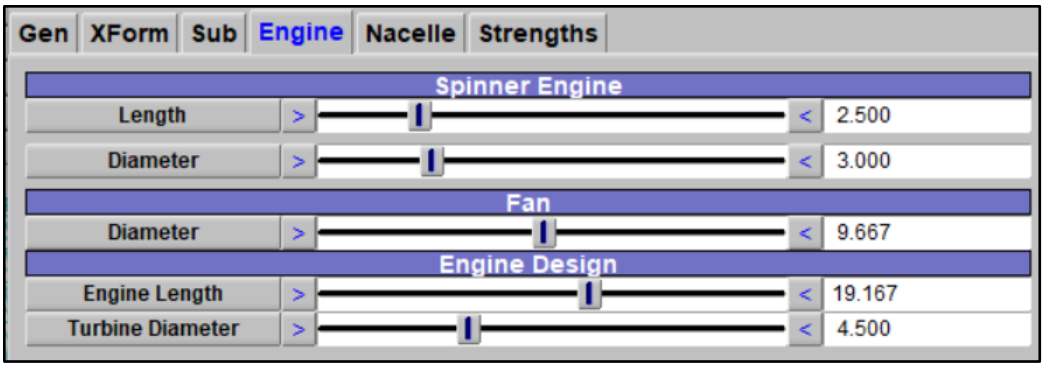


Figure 5. Slider setup of the Engine tab. *The configuration of the Engine tab can be seen above, along with the 5 sliders that allow for unique customization of the interior turbine portion of the model.*

B. Nacelle Tab

The next tab to the right is the nacelle tab which focuses on the shaping and geometry of the nacelle itself. In this tab, the user selects basic values to define the engine's inlet and body. For inlet design, this includes the length of the inlet, the diameter of the highlight cross-section, which is the leftmost cross-section in the engine model, and the scarfing of the front of the engine. This tab is also where the user gets to define the maximum diameter of the nacelle, how far the nacelle extends, and where in the *x*-direction this max diameter is located. Once the user has completed setting all of these values, they should have a fully defined engine and outer nacelle that is accurate and only took a minute or so to create.

The next components in the nacelle tab involve the fan and core exhaust areas, enhancing the model's specificity for CFD calculations. One of the main goals of this model was to surpass the mathematical accuracy of current design methods like flow-through engines and hollow tubes when it comes to CFD analysis. By allowing the user to define the fan and core exhaust areas, if the engine is uploaded into CFD software it will have fully defined boundary conditions for a dual stream engine. These area values can be found in specification documents or approximated and will provide more all-encompassing CFD calculations than a generic flow-through engine would. Additionally, two more sliders control the fan and core exhaust choke areas. Although the fan and core exhaust areas control the area of the surface right as the flow exits the engine, the choke sliders allow the user to specify the area of the exit as the air from the engine exits the entire model and enters the surrounding atmosphere. Figure 6 demonstrates the area of the engine model that each of these sliders control. Lastly, the remaining two sliders control the length from the fan exhaust face and the core exhaust face to each of their respective chokes.

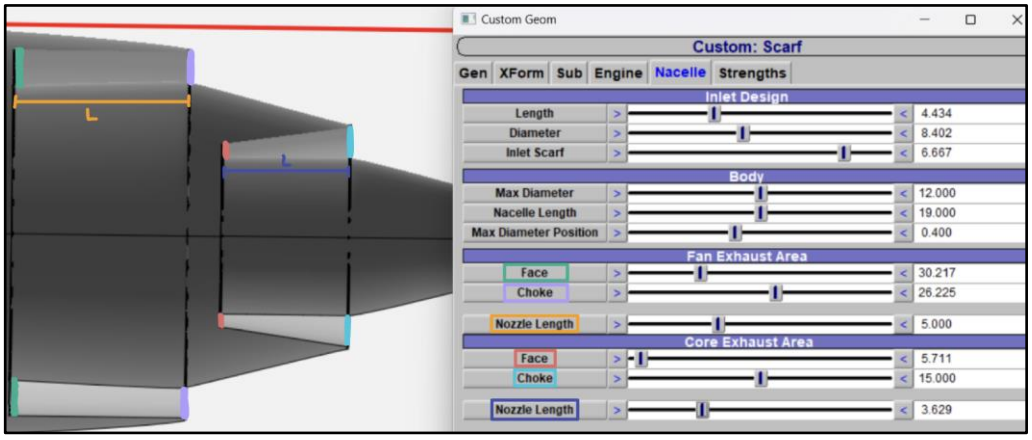


Figure 6. Diagram of engine exhaust with labeled sliders. *This figure provides clarification on which sliders control which portions of the rear engine. The lengths are defined by horizontal lines across the nozzles, and the areas are represented by color-coded vertical lines*

C. Strengths Tab

After defining the model according to the engine and nacelle tab, the engine should look very close to the desired outcome. The strengths tab helps the user perfect the shaping and curvature at the intersections of different cross-sections. As mentioned before, 'strength' is a term used by OpenVSP that defines the distance a line goes out from a cross-section before curving to meet the next cross-section. Normally, if a user were going through cross-section by cross-section or designing their own engine from scratch, strength is just a number that would be added onto the cross-section line. However, since it takes a lot of experience and trial and error to determine what numerical strength value creates the desired curvature, the slider format is beneficial. By controlling the strength values with a slider, a real-time representation of the strength can be observed, and a suitable value can be decided on easily. The strengths tab allows the user to have complete control over the shaping of the inlet, max diameter, spinner, and trailing edges of the engine model. This includes control over both the left and right edges of the stated cross-sections. The goal is that after going through the strengths section, the model will be precisely tuned to fit the needs of the user.

IV. Special Features

A. Subsurfaces

The first of several special features that will be covered is the ability for the model to be defined using subsurfaces. After fully defining the model using the engine, nacelle, and strengths tab, if it is a user's goal to do CFD calculations on the model, it will most likely be beneficial to add subsurfaces. This can be done by going into the analysis tab at the top of the OpenVSP application and clicking the 'CompGeom' tab. Doing this divides the engine model into predefined subsections (shown in Figure 7).

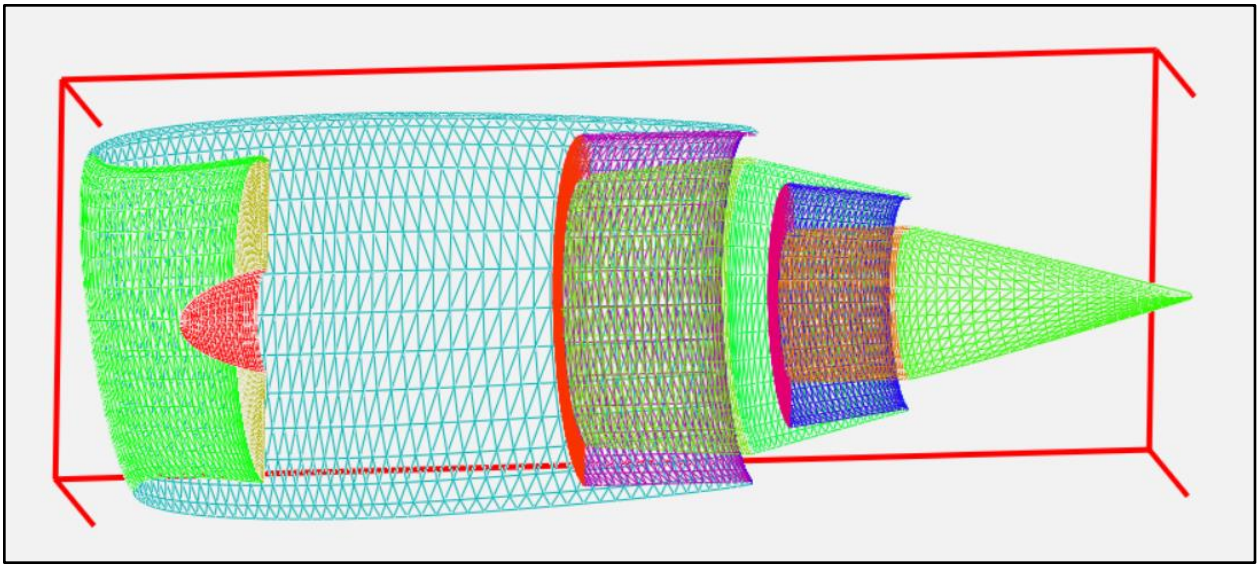


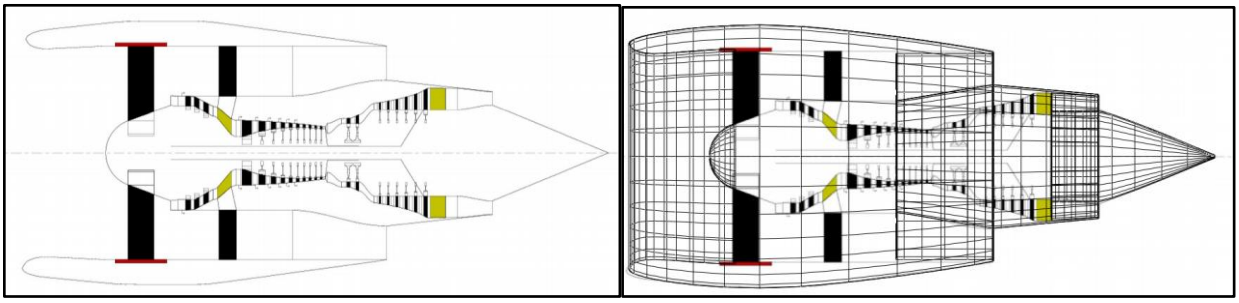
Figure 7. Engine model with defined subsections. *In Figure 7, each change in the color of the engine represents a new subsection. The subsections are numbered starting from the red spinner on the left and moving toward the right tip of the turbine plug. These subsections can later be called by number in CFD to define the boundary conditions of integral areas.*

The advantage of this is that there can be surfaces defined for the inlet, fan face, core exhaust, and fan exhaust. When the model with these defined surfaces is exported into a CFD solver, specific boundary and flow conditions can be defined for each subsurface. This successfully improves the engine model beyond a simple cylindrical representation where only inlet and exit conditions are defined. By having the core and fan exhaust surfaces defined separately, advanced calculations can be done on the model that will truly represent the realistic conditions of a dual-stream turbofan engine. Furthermore, having this engine model in OpenVSP makes it so that the subsurfaces

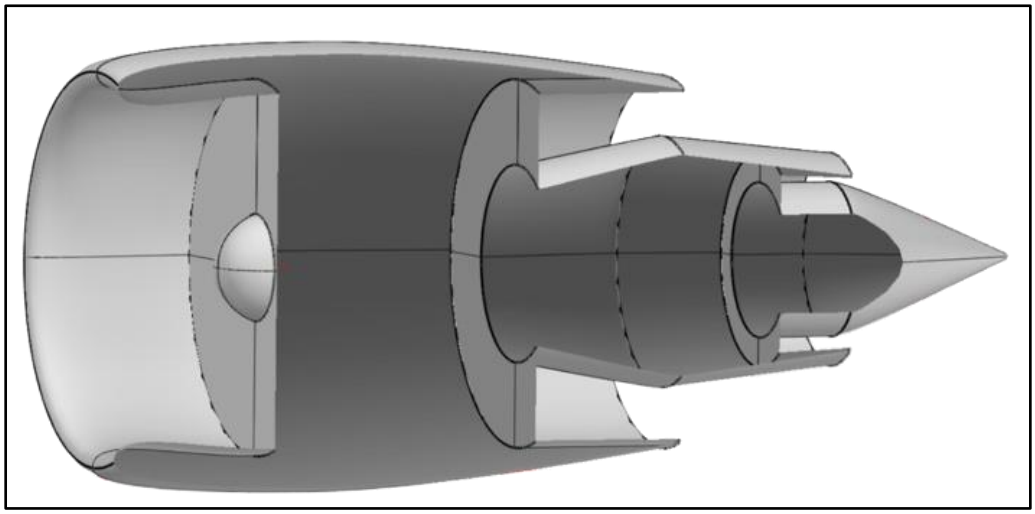
can be added with one click, simplifying the process of going through and defining individual surfaces. For example, the user can simply tell the CFD solver to use Subsurface 2 to represent the fan intake, Subsurface 6 to represent the fan exhaust, and Subsurface 10 to represent the core exhaust. These quick and easy steps, taking only a couple of minutes, can significantly boost the precision of the model and subsequent CFD calculations.

B. Overlay Background

One of OpenVSP's most advantageous features is the ability to upload an image into the background of the application. This becomes helpful because users can take a two-dimensional figure of an image, set it as the application background, change the engine component to mesh so that it is see-through, and then design the engine over top of the figure. This makes it simple to take any technical drawing of a turbofan engine and have a very close approximation of that exact model but in 3D form. This process can be seen below in Figure 8 where a GE90 Engine diagram from WATE++, as taken from Tong and Naylor's work, was modeled using the overlay background feature in OpenVSP [2].



a) Engine figure added to the background of OpenVSP [2] **b) Mesh model overlaying background figure**



c) Final engine after removing the background figure and changing the wire mesh to a shaded surface

Figure 8. Process of modeling an engine over a background image.

To use the overlay capability, the user must hover over the window tab in the top right-hand corner, and select background. Next, the box saying 'image' must be checked, allowing the user to import any engine figure or picture they would like onto the background of the page. This overlay feature acts as a proof of concept for the speed and ease of use of the customizable engine model. When using the overlay feature, order is crucial as some of the features have other parameters that they are dependent on. To make the modeling process as simple as possible the user should start at the nacelle tab and work their way to the right, finishing with strengths. The model does not have to be completely precise in the first two tabs as it is difficult to get all of the parameters correct without altering the strengths, which are in the last tab. If the model varies slightly from the background after going through the first two

tabs, it will most likely fix itself with the more fine-tuned controls in the third tab. This process can take a little getting used to, but the large amount of controllability is what makes this model beneficial. The overarching goal was to make it easier to add a more accurate engine component to OpenVSP and improve the overall model and any future CFD calculations. The sliders and visual aspect of the background overlay feature make it simple and intuitive for any engine design to be traced, even for those that do not have previous OpenVSP experience. The adjustable parameters are in terms that are well-known in the aerospace field and do not take extra research to put together. The entire process of uploading the image and adjusting all of the sliders of the engine can be done in a few brief minutes, and the outcome is a much more realistic and accurate engine than was available beforehand.

C. Scarfing

Lastly, another important feature that was made available by the customizable turbofan engine component is the scarfing feature (Figure 9). Scarfing refers to how, in some engines, the front inlet cross-section is not completely vertical, but instead slightly angled to reduce noise, optimize aerodynamics, and provide other advantages over an inlet that is completely perpendicular to the central axis. The angle of scarfing is controlled by the "Inlet Scarf" slider where zero represents no scarf, and when slid to the right, the increasing number corresponds to the angle (in degrees) the scarfed cross-section is tilted from vertical.

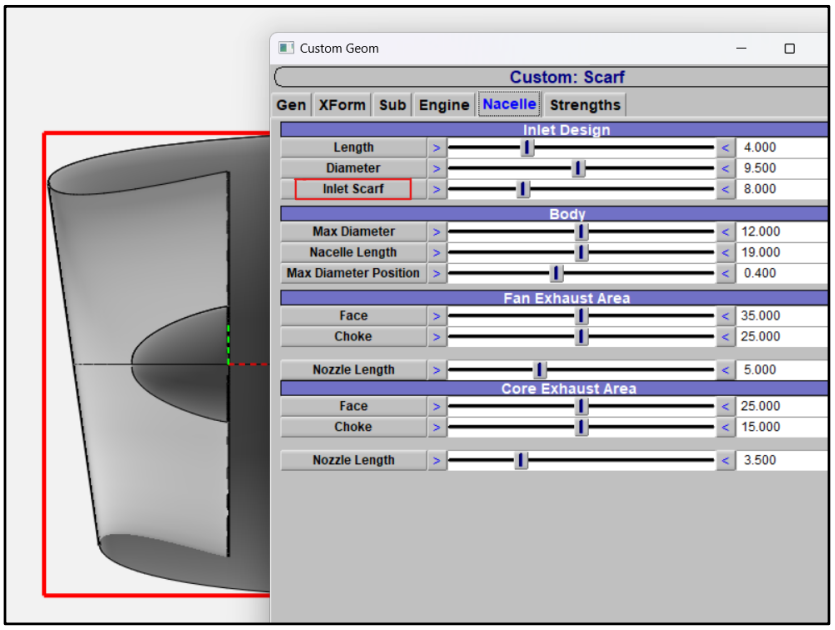


Figure 9. Demonstration of the engine scarfing component. *The scarfing of the engine above is controlled by the 'Inlet Scarf' slider in the Nacelle tab. The scarfing above is 8° from vertical according to the slider.*

The scarfing feature provides an advantage over simpler models that assume a completely axisymmetric design. A large portion of engines are non-axisymmetric and the ability to capture this in modeling is an important characteristic to have. Additionally, if someone decided to use a stack or a plain cylindrical model as a flow-through engine, adjusting a cylinder to have this precision could be a daunting task. By automatically building in the

capability to scarf the engine model, a complicated process was made a lot simpler, and a very common feature in engine design was made easily accessible.

V. Future Work

A. Inlet Design

One of the most important and complicated aspects of engines involves the design and shaping of the inlet. Though several techniques to parametrize the inlet were explored, none have been implemented yet. This component is mainly focused on the CFD application of the model as opposed to internal engine design. Because this project was not focused on the interior, mechanical features of the turbofan, shaping the inlet was one of the most important external parts of the model to focus on. The inlet must be designed in such a way that when air encounters the engine, flow separation and drag are minimized. If any of the angles on the inlet are too sharp or not designed with the correct curvature, then as soon as the angle of attack changes, the engine will no longer be optimized. As a result, inlet design was one of the more complicated portions of the engine and called for a lot of concentration. There are two main ways that a user can quickly design an adequate inlet using the model.

1. Elliptical Profile

The first option involves using a set of ellipses to shape the interior and exterior of the cowl. This would look something like what is displayed in Figure 10, with the red circle shaping the exterior cowl lip (the portion of the cowl above the yellow line) and the blue ellipse shaping the interior cowl lip (below the yellow line). Oftentimes, the upper cowl is guided by an ellipse with an aspect ratio of 1:1, and the lower cowl is guided by a ratio of 2:1. Frequently inlets are defined by these kinds of elliptical profiles, and while the aspect ratios could change based on the model, the concept stays the same. OpenVSP uses strengths to define curvature, meaning that the user controls how far a line goes out at a certain angle before it curves. This makes the inlet modeling process somewhat difficult, as it is hard to look at a certain numerical strength value and ensure it accurately represents a correct curvature.

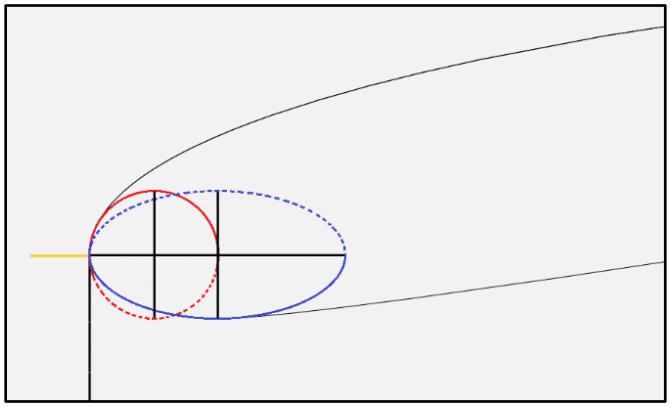


Figure 10. Two elliptical cross-sections define the inlet. *In Figure 10 the red (1:1) and blue (2:1) elliptical cross-sections define the upper and lower cowl curvature of the inlet*

Given this, the easiest way to get the inlet to follow an elliptical profile is by importing two stacks (cylindrical components defined by cross-sections) into the model and configuring them so that they consist of a single elliptical cross-section. The ellipse can then be moved to the highlight diameter by going into the 'XForm' tab, turning on UW, translation, and adjusting the U-coordinates until the ellipse is centered around the highlight diameter cross-section (shown in Figure 11). Then, by adjusting the *x*-location to a positive value equal to the semi-major axis, or half the width of the ellipse, the stack will be positioned precisely at the edge of the inlet.

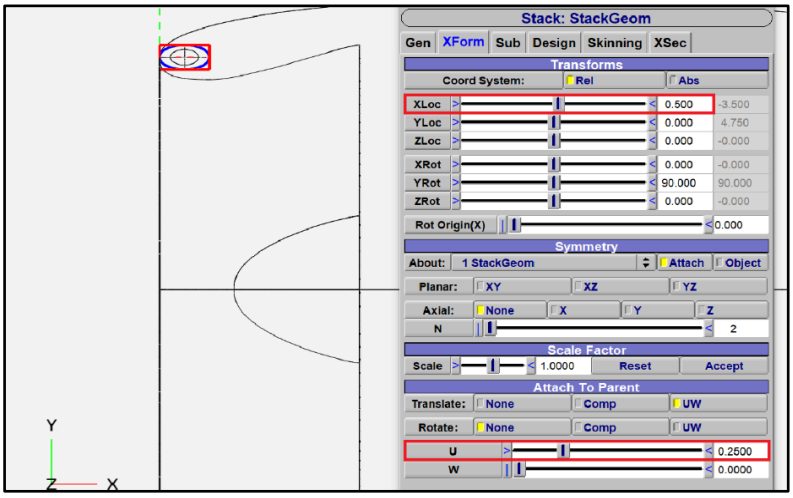


Figure 11. Method of positioning elliptical stacks on the inlet correctly. *The coordinate system of this stack is positioned at the left edge of the highlight cross-sections, by adjusting U and the XLoc sliders, the ellipses can be positioned such that they look like Figure 10.*

The same process can be used for another stack, and the aspect ratio can be adjusted so that the size of the ellipse creates an inlet shape that is appropriate for the engine. Using the ellipses as a guide to follow, the user can then adjust the inlet using the parameters under 'Inlet Design' in the nacelle tab and inlet parameters in the strengths tab. Through these steps, an accurate inlet mold can be created for the engine model, and the geometry should respond positively to CFD.

2. NACA-1 Profile

The other method that is oftentimes used to define the inlet is to have the inlet follow the curvature of a NACA-1 airfoil. The NACA-1 airfoil is already suitable for the aerodynamics of an aircraft in flight, and by utilizing it on the engine, drag can be minimized. The NACA-1 curving of the inlet was originally done by plotting a curve through a series of points that corresponded with the airfoil. Later, it was found that this shaping could be formed by typing in a specific number for that engine that represented the inlet curvature when it was set to follow the NACA-1 geometry.

Through conversations with an inlet designer, it was found that the most common way of defining an inlet would be to use the elliptical profiling on the interior cowl, or underside of the inlet, and the NACA-1 pattern on the exterior cowl [3]. In future work, a way to implement this design properly could be found. Since there are many various ways to define an inlet and a lot of them depend on user preference, it was difficult to decide on a concrete way to simplify this portion of the engine. One idea was to have a slider where if the user drags it to one side, the inlet takes on the shaping of a NACA-1 profile, if it is dragged to the other end, it follows an elliptical shape with a 2:1 ratio, and in the middle, the OpenVSP strengths remain, and users can adjust the curvature of the model precisely to their liking. This technique could be implemented in future versions of the model.

B. Computational Fluid Dynamics

An ideal future addition to the engine is the ability to quickly import the model into a CFD solver. In theory, integrating the engine component with existing CFD export techniques for OpenVSP should be a smooth process. However, this has yet to be tested due to a lack of access to CFD software and minimal exposure to various CFD solvers. However, the customizable turbofan engine component has been developed with this capability in mind, even if it is not yet implemented. Furthermore, the layout and capabilities in OpenVSP, such as subsurfaces that allow easy section breaks for the definition of boundary conditions, should make the process of importing the engine into a CFD solver smooth and simple. There should not be a lot of work necessary to make this feature functional.

C. User Input

Though the customizable turbofan engine component is certainly an improvement to the current methods used to quickly design and model engines, the real improvement will come from user input. The most difficult part of designing this model was deciding which parameters the user should have control over, and which parameters should be controlled behind the scenes. In addition, as mentioned at the beginning of the paper, choosing how variables would be controlled, for example, area or diameter, was a difficult decision to make. The only way to truly know what works best is to get the model out to the public and allow users with different levels of experience, backgrounds, and projects to decide what works best for them. By understanding what kind of preferences a wide range of people might have, the model can be better customized to fit the general needs of its users.

That being said, this model is a starting model intended to be a beta release. It is not perfect, and will not fit the exact needs of everyone who uses it, but it will help lay the groundwork for future development. It is the hope that users will leverage this component to make their models slightly easier and more accurate while providing feedback, and, if desired, customizing the code themselves. OpenVSP's open-source nature allows users to easily modify the CustomScript file of the engine component to suit their specific needs, while providing valuable feedback for future improvement.

VI. Concluding Remarks

Prior to this customizable turbofan engine component, OpenVSP lacked the capability of importing and parametrically adjusting an engine component. Consequently, users often omitted the engines from their models or used simple cylindrical flow-through engines for CFD calculations. This limited the accuracy of the calculations and aircraft models severely. However, the process of making an accurate turbofan engine involved going through a long and cumbersome process of developing the model cross-section by cross-section. The goal of this project was to streamline that process and limit the long, drawn-out design that was currently required. The customizable turbofan engine strikes a balance between designing an engine from scratch and using a completely pre-defined, non-customizable component. Through the implementation of this engine component, fully modeling a complete engine or aircraft is both accelerated and simplified. Additionally, this model can be overlaid on top of a 2D figure and a 3D replica of an engine can be created. The code for this model is published on GitHub where it will hopefully be used, modified, and made better by a broad range of users. This collaborative effort ultimately aims to make the process of modeling and conducting CFD calculations easier and more precise.

Acknowledgments

The author thanks Nat Blaesser of NASA Langley Research Center for guidance and mentorship and John Slater for his discussions on inlet design. This work was done as part of an unfunded mentorship through the Governor's School for Science and Technology in Hampton, VA.

References

[1] Robert A. McDonald and James R. Gloudemans. "Open Vehicle Sketch Pad: An Open Source Parametric Geometry and Analysis Tool for Conceptual Aircraft Design," AIAA 2022-0004. AIAA SCITECH 2022 Forum. January 2022.
[2] Tong, M.T., & Naylor, B.A. (2008). An Object-Oriented Computer Code for Aircraft Engine Weight Estimation.
[3] John W. Slater. "SUPIN: A Computational Tool for Supersonic Inlet Design," AIAA 2016-0530. 54th AIAA Aerospace Sciences Meeting. January 2016.