Automated Training Data Generation for Geometrical Machine Learning

Team: nullPtr
Allen Jiang, Bryant Porras, Kuanren Qian, Shu You, Vimalesh Vasu,

Introduction and Background:

Robotic post-processing of metal 3D prints aims to automate the cleaning of printed parts when they are initially embedded in a bed of metal sand. The first step in this process is to vacuum the metal sand efficiently with a vacuum nozzle [Figure 1] optimized for metal sand intake. Given the physical constraints of this unique system, it is too complicated for any current CAD solver to produce an optimal design through its own optimization software. Therefore, the current method of determining an ideal shape for such a nozzle is iterative parametric optimization.

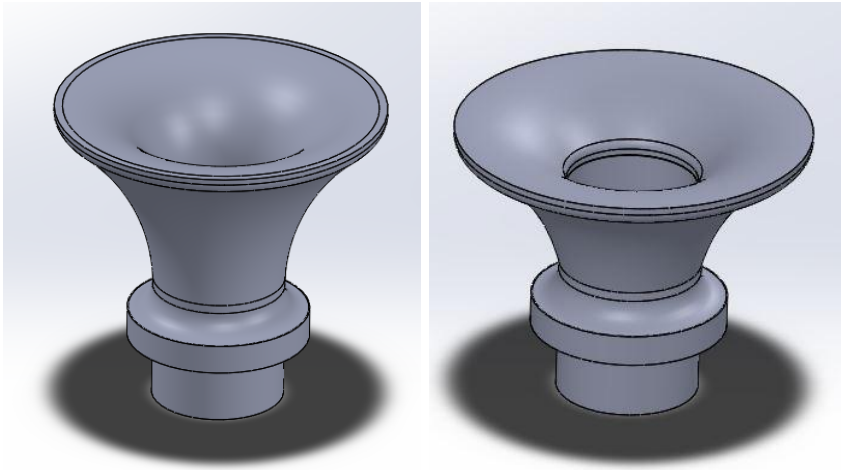

Figure 1: Two parametric variations of the same generalized vacuum nozzle design

Iterative parametric optimization of the nozzle shape is a multi-step process that involves creating an initial generalized design, followed by running a fluid simulation to obtain the velocity and pressure profiles of the design [Figure 2], and then adjusting the design parameters based on the faults realized in the profiles. The high-level goal is to iterate through these steps until an optimal design is reached.

However, given the tedious and time-consuming nature of running fluid simulations, such a high-level task would take an exorbitant amount of time and mindless labor. Simply put, fluid simulations require a large number of preprocessing steps, such as defining environments, materials, points of contact, and so on. However, for parametric changes that do not alter the basic general design of the part [Figure 1], such preprocessing is identical for each iteration. To ease this situation and take advantage of this monotony, we propose a C++ program that would run parametric variations of nozzle designs through the same fluid simulation to automate this tedious process. Such a batch simulation would run through a large set of geometries in one

program run and output all simulation results for a user to judge which geometry is best qualitatively.

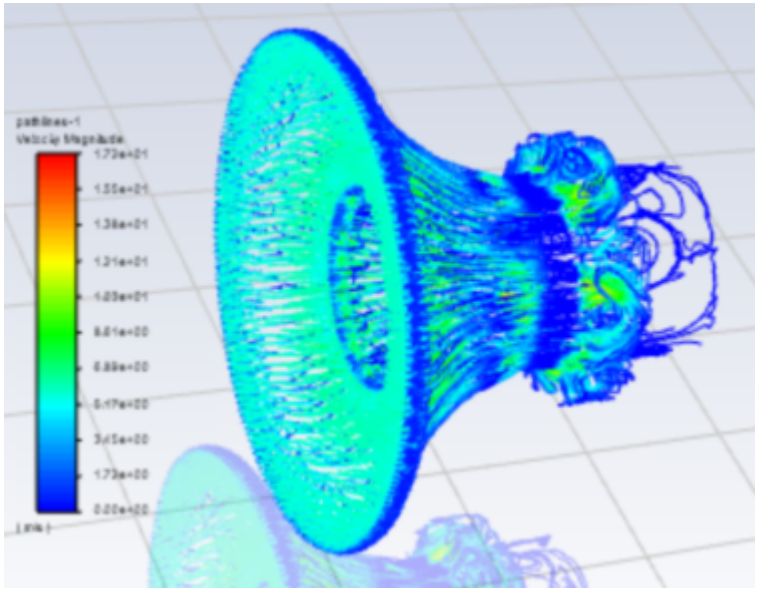

Figure 2: Ansys fluid simulation outputting the velocity profile of a vacuum nozzle design

Project Goal:

The goal of the project is to develop a C++ program capable of automating simulation procedures and communicating between different commercially available software to generate training data for machine learning algorithms. The proposed plan will also visualize batch simulation results in a simple GUI to inform and assist users in making optimal design decisions. We aim to tackle the target physics problem of computational fluid dynamics (CFD) simulation of vacuum nozzles in batch. There are several issues to tackle: (1) Geometry and/or mesh generation based on parametric user input. (2) Simulation convergence. (3) Batch simulations using commercial software. (4) Processing and visualization of extensive simulation data. All of the above issues are discussed in detail in the solution section.

Solution and Approaches (Technical details):

Task 1: Automated geometry/mesh generation.

In this task, we aim to automate the mesh generation process for vacuum tubes of various profiles. In CFD analysis, boundary layer mesh is necessary to achieve a converged solution. However, boundary layer mesh poses significant challenges when dealing with geometries with complex features, such as fillets. We will first generate a simplified 2D vacuum tube profile in 2D space to tackle this problem without detailed features. Then, we will use a predetermined mesh template based on the specific vacuum tube through scaling and translating of nodes in 2D space. Finally, we will rotate all the nodes around an axis and generate a 3D mesh ready for simulation.

Task 2: Parameter tuning for simulation convergence

We will use a standard vacuum tube geometry to conduct CFD simulations using commercially available software in this task. We plan to test different simulation parameters and material properties based on targeted vacuum tube operating environment conditions to find a reasonable range of parameters as input. This task will allow us to generate input files that can successfully converge without failures when running batch simulation.

Task 3: Batch simulation scripts for commercially available software

In this task, we will develop a component in our program to automatically generate batch simulation scripts that commercially available software can read and execute. The expected outcome of this task is to make our program call commercially available software and pass batch simulation script as input.

Task 4: Batch simulation results visualization in GUI

We will develop a simple GUI in our c++ program to visualize batch-generated simulation results from commercially available software in this task. We will use existing GUI libraries to simplify visualization and develop a component that can display 3D contour plots based on obtained results. The expected GUI component will display results side-by-side to provide a better comparison analysis for users.

Fallbacks:

Task 1: Automated geometry/mesh generation.

Depending on the difficulties of the problem we will encounter, we will simplify the vacuum tube geometry to avoid complex features. If we face significant troubles when generating complex 3D mesh with correct element connectivity information, we will abandon 3D mesh and switch back to 2D mesh. We will conduct axis symmetry simulation in commercial simulation software instead.

Task 2: Parameter tuning for simulation convergence

If simulation results from commercially available software show that our targeting simulation conditions will lead to unavoidable divergence, we will modify our expected simulation condition accordingly to achieve good convergence. Further studies on convergence on different simulation conditions can be conducted if the primary target of this project is met.

Task 3: Batch simulation scripts for commercially available software

Considering the availability of different simulation software, we will switch between ANSYS and Abaqus depending on command-line control complexity.

Task 4: Batch simulation results visualization in GUI

If we encounter difficulties in the time frame of this project and consider task 4 as the last component of this project, task 4 may be omitted and commercially available software will be used for visualization.

Milestones:

As described in the project file.