Automatically Generating 60,000 CAD Variants for Big Data Applications

Satchit Ramnath, Payam Haghighi, Ji Kim, Duane Detwiler, Michael Berry, Jami Shah, Nikola Aulig, Patricia Wollstadt, Stefan Menzel

2019

Proceedings of the ASME 2019
International Design Engineering Technical Conferences
and Computers and Information in Engineering Conference
IDETC/CIE 2019
August 18 – 21, 2019, Anaheim, CA, USA

DETC2019-97378

AUTOMATICALLY GENERATING 60,000 CAD VARIANTS FOR BIG DATA APPLICATIONS

Satchit Ramnath[1], Payam Haghighi[2], Ji Hoon Kim[2], Duane Detwiler[3], Micahel Berry[3], Jami J. Shah[2], Nikola Aulig[4], Patricia Wollstadt[4], Stefan Menzel[4]

[1] Simulation Innovation & Modeling Center, The Ohio State University, Columbus, Ohio, USA
[2] Digital Design & Manufacturing Lab, The Ohio State University, Columbus, Ohio, USA
[3] Honda R&D Americas, Raymond, Ohio, USA
[4] Honda Research Institute Europe, Offenbach/Main, Germany

ABSTRACT

Machine learning is opening up new ways of optimizing designs but it requires large data sets for training and verification. While such data sets already exist for financial, sales and business applications, this is not the case for engineering product design data. This paper discusses our efforts in curating a large Computer Aided Design (CAD) data set with desired variety and validity for automotive body structural compositions. Manual creation of 60,000 CAD variants is obviously not viable so we examine several approaches that can be automated with commercial CAD systems such as Parametric Design, Feature Based Design, Design Tables/Catalogs of Variants and Macros. We discuss pros and cons of each method and how we devised a combination of these approaches. This hybrid approach was used in association with DOE tables. Since the geometric configurations and characteristics need to be correlated to performance (structural integrity), the paper also demonstrates automated workflows to perform FEA on CAD models generated. Key simulation results can then be associated with CAD geometry and, for example, processes using machine learning algorithms for both supervised and unsupervised learning. The information obtained from the application of such methods to historical CAD models may help to understand the reasoning behind experiential design decisions. With the increase in computing power and network speed, such datasets together with novel machine learning methods, could assist in generating better designs, which could potentially be obtained by a combination of existing ones, or might provide insights into completely new design concepts meeting or exceeding the performance requirements.

1. INTRODUCTION

Automotive body structures have evolved through decades of vehicle generations with the objective to, one the one hand, satisfy a multitude of safety requirements and, on the other hand, to minimize weight and cost. To improve designs from model year to model year, data from crash testing, simulations and service in the field is used to improve designs. However, the design of functional components, for instance, A, B, and C pillars, is highly constrained by styling surfaces, packaging requirements and closures such as doors and windows. To improve designs under these constraints, structure designers tweak an existent component's geometry by trial and error to incorporate new data and apply required changes from model to model. New designers are thus heavily dependent on existing designs used in the past. In modern design processes, there is a desire to move away from the preconceived notion of what designers have considered "good enough" in the past and to make use of novel, especially computer-aided techniques, in the design of automotive components.

The rapid development of the topology optimization and the availability of commercial tools offers the possibility to find load paths, from which to derive conceptual designs as starting point [1][2]. However, the applicability of topology optimization to the automotive design process is limited: tools produce monoliths of solid cross sections, whereas automotive body components are mostly sheet stamped pieces joined together with spot welds or other hybrid joining methods. While there is ongoing research in both "in process" and "post process" methods [3][4] to produce multi component assemblies of thin walled components, the method is not able to handle the details required for real world components. Another obstacle is that multiple linear and nonlinear load cases are involved in evaluating performance of automotive structures; combining multiple nonlinear load cases in FEA results is theoretically non-tractable.

Due to the complexities an engineer is faced with, efficiency of the virtual development process may be improved by support from machine learning or artificial intelligence systems which are immensely successful in, for instance, natural language processing or image recognition. A comparable system applicable to engineering designs would have potential to harness all the data that is available from previous developments in a way similar to the engineers' experience, but with complementary skills such as making sense out of big amounts of previous data on designs, performance and decision consequences.

Recent approaches on geometric deep learning [5][6] extend the principles of artificial neural network methods towards handling 3d geometries, which are the primary data type found in the automotive design process. For example, so-called autoencoder architectures for 3D objects [7][8] facilitate new computational representations for geometries and have successfully been applied to topology optimization [9]. Furthermore, machine learning or optimization approaches may be applied to state-based representations, based on the structural state of a design under load [10][11].

However, novel machine learning and artificial intelligence methods require a high amount of data for training, while typically the geometric and performance data of industrial development processes are not stuctured in a database suitable for application of the methods mentioned. Therefore, in this paper we present a method to generate a suitably large dataset that can be used as starting point for research on bringing computer-aided engineering together with state of the art machine learning and artificial intelligence approaches.

1.1 Set Based Design and Optimization

Up to now, there are several practical relevant approaches for design optimization in virtual vehicle development. Traditional mathematical methods such as hill-climbing algorithms are referred to as point based because they work with a single design point which is incrementally improved. They are mostly single objective, highly dependent on the start point, and in danger of stalling at local peaks. On the other hand, set based methods exist, which work with several data points. For global optimization, stochastic search algorithms are proposed, such as genetic algorithms (GA) or evolution strategies (ES) [12]. Both methods utilize an iterative semi random approach to improve populations of parameter/feature variations. For these optimization methods, the number of generated samples to find an optimal solution is not known beforehand. In contrast, sampling methods, such as Design of Experiments (DOE) and Taguchi, allow to specify beforehand the number of parameter/feature combinations to evaluate since they are driven by some sampling scheme, e.g. latin hypercube sampling. Based on these data samples, a response surface or surrogate is constructed, e.g. kriging models, regression models or neural networks, and an optimization is carried out on the surrogate with eventually re-evaluation on the real performance function. Of course, modern methods also integrate surrogates in evolutionary optimization to speed up the overall runtime by partially evaluating populations on the surrogate. However with an increasing number of parameters/features it is difficult, if not impossible, to get response surfaces that fit the data well. The number of parameters/features is typically specified based on human heuristics or on data analytics, e.g. sensitivities. Recently geometric deep learning [5] or spectral decomposition of geometries [13][14] are new data-driven approaches to find abstract geometry-representing features for 3D designs in the network latent representation or the spectral coefficients.

1.2 Parametric and Feature Based Methods in CAD

Most CAD systems support a range of parametric design methods [15], which include 2D constraints, user defined features, design tables or catalogs, user-defined macros and parametric equations.

2D Constraints: A large number of 3D features are sketch based (extrude, revolve, loft, general sweep etc.). Sketch entities can be dimensioned and constrained so that only a few parameters drive the geometry. Even a fully constrained sketch can have multiple solutions due to the nonlinear nature of constraint equations. In such cases, CAD systems use heuristics to choose a particular solution. Also, there may be no valid solutions for certain parameter values (e.g. two sides of a triangle must be greater than third). One other issue is that if there are inner loops in a sketch, certain combinations may cause self-intersections of the sketch or change the number of loops – all of these will lead to failure in generating the 3D feature from that sketch.

Design Catalogs/Tables: Provides a unified parametric scheme for an entire component. Their use is in creating one CAD model for size variants. Tables lack variety, even if a CAD system allows selective activate or suppression of localized features.

User Defined Features: Combine parameters from sketches with 3D parameters (e.g. sweep length, direction), as well as feature sequence. This amounts to adding a sub-tree in the construction history. They provide a mechanism to add a localized shape. One issue is positioning these features relative to other features in a robust way. This is easier interactively, as reference entities can be picked by the user. If the features are sketch based, it brings along all the issues mentioned for sketches and 2D constraints.

Macros: Macros encode a geometry creation procedure. It could apply to creating an entire part or selected features or modifications. Macros can support conditional actions, iterations, nested loops, giving them greater flexibility.

Parametric Equations: Parametric equations can be used to cross link parameters between different levels. For example, 2D sketch parameters related to 3D feature parameters; parameters in multiple sketches related; dimensions driven by functional requirements such as allowable stress.

In Section 2 we will consider the suitability of each of these for automatically generating large data sets.

1.3 Big Data & Artificial Neural Networks

Recent advances in computational capabilities to generate and store large amount of data have led to the increasing availability of so-called big data. The term big data refers to data set sizes that cannot be handled by commonly used commercial software and methods for data mining, analysis and machine learning. A definition of big data is given as, "*Big data represents the information assets characterized by such a high volume, velocity, and variety to require specific technology and analytical methods for its transformation into value*" [16]. There are three important characteristics of big data known as the three Vs - Volume, Variety, and Velocity [17]. Volume refers to the quantity of data generated and stored,

Variety is the type and nature of data in the dataset, and Velocity is the time taken to generate the dataset.

Artificial Neural Networks (ANN) consist of an input layer, an output layer and may comprise several connected interior (hidden) layers of neurons. Hidden layers produce an output from inputs and embedded functions that are learned from training data by adapting weights and biases [17]. Network architectures with a high number of hidden layers are commonly referred as deep neural networks and have recently provided impressive results in a variety of application domains [18]. They, however, require large amounts of training data for successful application. ANNs have traditionally been used for regression or classification tasks. For the latter, ANN may classify data samples into two categories (e.g., structure meets or does not meet criteria; Good/NoGood; fraudulent or non-fraudulent transaction), or multiple groups (e.g. risk level, Measure of Goodness Rating on some scale). For a Boolean performance output (G/NG), there will only be two neurons in the output layer (Figure 1) [10]. The input will be either raw data or predefined features. Training of an ANN means the processing of training, while for each sample or a batch of samples, the (randomly) initialized weights are adapted such that some error or loss function is minimized.

There are many open source and commercial packages available for deploying ANN deep learning, such as Matlab NN Toolbox [19] for standard network architecture and Google Tensor Flow [20] for deep neural networks.

Big data plays an important role in shaping various aspects of mechanical engineering like product design and development, product manufacturing and product efficiency [21]. Data are collected over the product design and development process, and also during the Product life cycle (PLC). Thus, a designer's knowledge and experience along with customer feedback are incorporated into the data collected, such that data mining techniques offer the opportunity to innovate and create new products by facilitating information visibility and process automation in design and manufacturing [22].

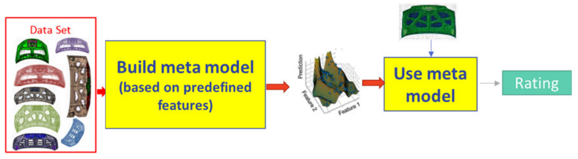
There are different types of data that can be mined for various objectives. For example, in product design and manufacturing data mining may help to manufacture better next-generation products, sharpen competitive advantage and reduce the overall product lifecycle time [23]. One type of data that can be mined for optimization is the CAD model of a product's geometry from the conceptual design stage. State-of-the-art machine learning and ANN models offer the possibility to learn from geometric data, for instance, to solve classification tasks for geometric shapes or to learn features in an unsupervised fashion as discussed by Achlioptas et. al. in [7]. Especially unsupervised feature learning is a promising approach for the extraction of novel features that are commonly not considered in traditional computer-aided design [24].

1.4 Scope of Study & Functional Requirements:

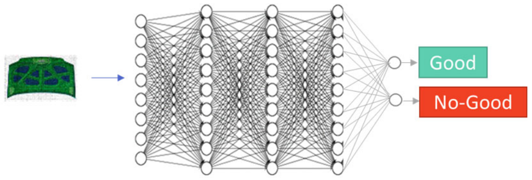
Although the approach taken in this project can be applied to any structural component, the scope of this project is for the structure of automotive hoods. These components must meet several performance requirements: in the case of an automotive hood, hood lift and twist deflection (linear static analyses); pedestrian impact and frontal impact (dynamic and non-linear analyses). Since these structures are made from sheet metal stamped components, stiffness is achieved by creating ribs and channels, while light weighting is achieved by cutouts. The hood frame, which is responsible for almost all of the stiffness and strength, is bonded to the skin (class A surface) with adhesives. The size, aspect ratio and curvature of the skin varies from vehicle to vehicle which constrains the overall frame structure.

The big data set needed for supervised or unsupervised learning must meet the following functional requirements:

- Automatically generate CAD models
- Automatically generate mesh and apply boundary conditions (BCs) to finite element (FE) models
- Models must be geometrically valid
- Models must meet manufacturability criteria
- Models must produce a wide variety of feature shapes, sizes and pattern constraints



a) Response surface approach



b) Deep Learning Approach

Figure 1: Correlating performance with geometry,
(a) Traditional approach, (b) Multi-layer neural network

In this paper, we discuss our approach to generate a large amount of 3D CAD data, using CATIA v5, with the purpose to enable various machine learning and data mining tools to generate new designs and facilitate the creative design process. The conceptual designs (CAD models) can also be fed to a FEA solver to determine performance metrics. For this purpose, the paper first discusses different approaches to create large amount of data followed by discussion on step-by-step process with a case study. This method is feature based, and the designer is required to add any new feature to a collection of features called the feature library. Thus, the variety of designs achieved for the machine learning algorithm is affected by the variety of features present in the library.

2. RESEARCH METHODOLOGY FOR CAD DATA GENERATION

Although the methodology proposed here could be implemented with any major CAD system, we used CATIA V5 as it is used widely by big auto companies, such as Ford, Honda, Tesla etc. As part of this research, two general methods/tools to automate the generation of explicit CAD models were proposed:

1. Implicit model (unevaluated geometry): Construction history, generative shape design operations, procedures and sequence
2. Explicit Model (evaluated geometry): Identification of geometric features and their parameters, distinguish between relevant and irrelevant features

We began by dissecting existing hood design, to understand the construction history, but they contain hundreds of operations, and car model specific features such as seal beads, bolt holes, and character line which are not relevant to this study. The general procedure of creating a hood frame along with the hood skin for mass production cars is shown in figure 2.

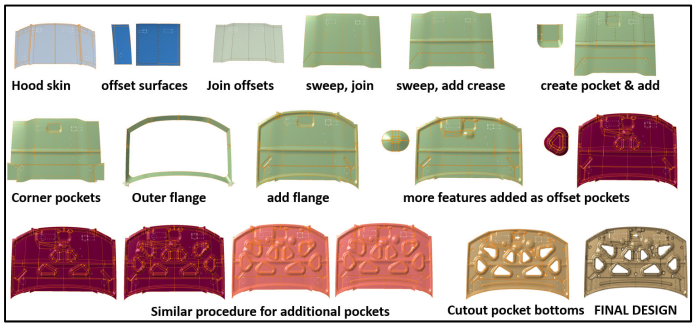


Figure 2: Overall procedure of the design of a hood

So, idealized models of hood frames were extracted by only considering the main structural features and disregarding the features that do not play a role in structural integrity of the model. A few idealized hoods are shown below in Figure 3.

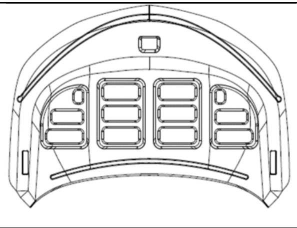
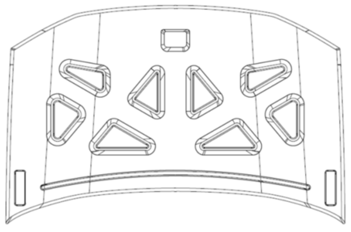
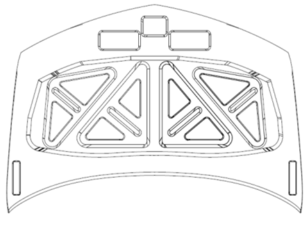
Model #	Idealized hood Sketch
1	
2	
3	

Figure 3: Idealized hood examples

Idealized hood models were defeaturized and key features such as the rib patterns, pockets, outer shape and profile of side, rear and front ribs and connections such as hem, mastic beads and bolt locations were identified on hood frames. A few of them are displayed in the table below.

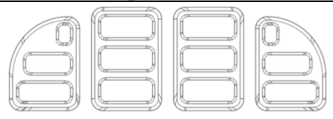
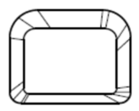
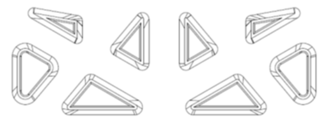
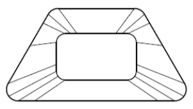
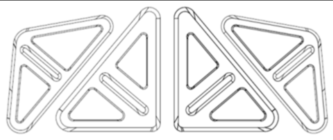
Main ribs with pockets	Hood lock flats
	
	
	
Front/rear rib	Hinge flat

Figure 4: Some of the key features identified on the hood frame

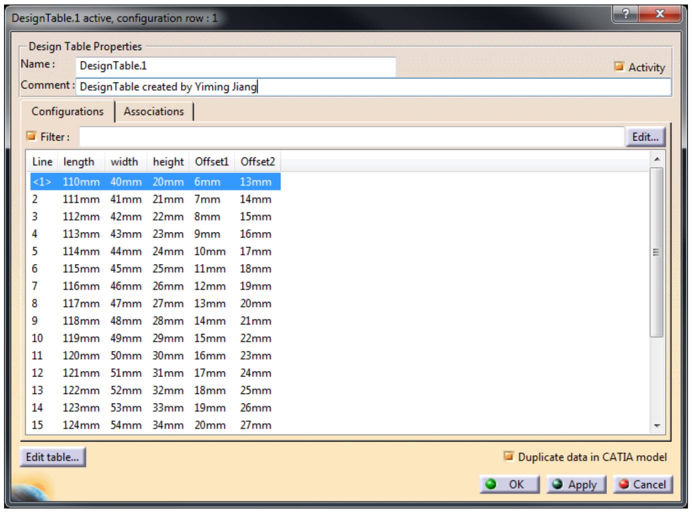
As part of this research, multiple methods/tools to automate the generation of explicit CAD models are compared. The methods investigated are:

1. Design/Parameter Tables
2. User Defined Features with Design Tables
3. Macros with Parameters Table

Each one of these methods are discussed in detail in following sub-sections.

2.1 Design/Parameter Tables:

Design table is a parametrizing tool within CATIA to create and manage component families [25]. Components such as parts can be parametrized and stored as a table, using Microsoft Excel or in a tab delimited text file. Shown in figure 5, the structure of a design table in CATIA where, columns represent all the parameters in the part, while rows define the set of values for each variation. By loading the part file defined in the design table, the parameters are updated accordingly. This saves the time and effort for the designers to generate each part individually. Figure 6 shows an example of a parameterized sketch with features and size parameters for the boundary generated using the design table: features are easily varied according to the dimensions defined in the design table. While such advantage exists, the designers have to make sure to define all required constraints, and make sure the values in the design table do not generate non-manifold designs. Thus, the designer should be aware of the ranging limits of the parameterized dimensions. Additionally, the units of the dimensions have to be consistent in order to maintain integrity of the models. However, such guidelines are essential for any parametric modeling.



DesignTable.1 active, configuration row : 1

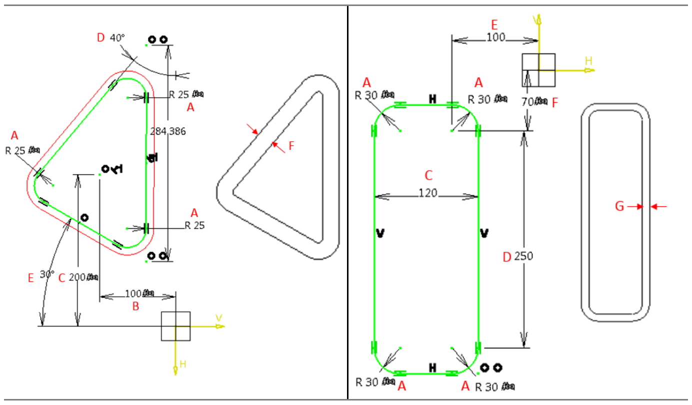
Design Table Properties

Name : DesignTable.1

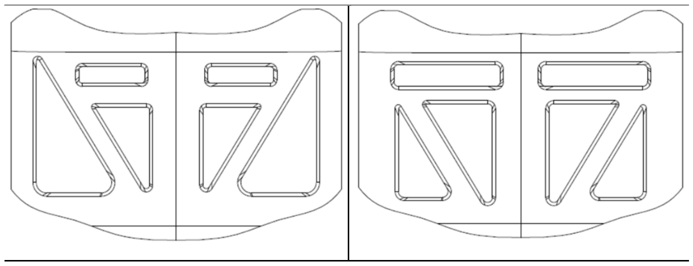
Comment : DesignTable created by Yiming Jiang

Line	length	width	height	Offset1	Offset2
<1>	110mm	40mm	20mm	6mm	13mm
2	111mm	41mm	21mm	7mm	14mm
3	112mm	42mm	22mm	8mm	15mm
4	113mm	43mm	23mm	9mm	16mm
5	114mm	44mm	24mm	10mm	17mm
6	115mm	45mm	25mm	11mm	18mm
7	116mm	46mm	26mm	12mm	19mm
8	117mm	47mm	27mm	13mm	20mm
9	118mm	48mm	28mm	14mm	21mm
10	119mm	49mm	29mm	15mm	22mm
11	120mm	50mm	30mm	16mm	23mm
12	121mm	51mm	31mm	17mm	24mm
13	122mm	52mm	32mm	18mm	25mm
14	123mm	53mm	33mm	19mm	26mm
15	124mm	54mm	34mm	20mm	27mm

Figure 5: Interface of design table used in CATIA



a) Sketches of the parametrized features



b) Instantiated features with different parameter values

Figure 6: Example of parametrized features using design table

This method provides the ease of integrating and generating CAD designs in CATIA V5. However, it is not always possible to integrate/combine individual features generated from a design table, while applying external constraints like manufacturing constraints, to generate a fresh design, given the number of models required to create a large dataset. Other commercial systems like SIEMENS NX®, uses *expressions* to read parameter values form a design table and feed it into the parameters in the part.

2.2 User Defined Features with Design Table for Parameters:

Power copies are a group of CATIA entities (eg. points, lines, surfaces, formulae etc.) generated in a particular order, that are independent of the environment they were originally created in, and can easily be adapted to a new environment. The concept of creating power copies is similar to part families in that, it contains the reference to an external model, driven by a set of pre-defined parameters. However, the power copies generated need not necessarily represent an entire part, but can represent only a certain feature.

Primary entities required to create a power copy are *Definitions*, *Inputs* and *Parameters*. The terms are defined as follows:

Definitions: The geometries or entities generated when a power copy is instantiated

Inputs: The geometrical entities that serve as an input to the power copy

Parameters: The driving parameters used in the generation of the geometry, using a power copy

In the process of creating power copies, the construction methodology can be captured either for a single feature or for multiple features. Creating power copies for multiple features are called *Compound Features*.

Similar to the previous method of using just the design table, generating hundreds of thousands of designs using power copies would take immense manual effort since the power copies have to be instantiated for each row in the design table.

2.3 Macros with Parameters Table:

The architecture of CATIA called the *CATIA Application Architecture* (CAA) consists of *Application Programming Interfaces* (APIs) used to develop user specific functionality for CATIA. The three primary ways to create programs that interact with CATIA are *VBScript, VBA* and *Visual Basic*. CATScript is a scripting language, in VBScript, developed specifically for CATIA. The advantages of CATScript are the ability to record a series of operations performed in the CATIA window and the ability to access CATIA's pre-defined form features (eg. extrude, sweep etc.) directly.

The data structure for CATIA v5 automation consists of these main components, namely, ***Collections***, and ***Objects***. Collection is a group of similar objects. Objects (e.g. pad, pocket, sketch, line, etc.) are entities created, selected and/or modified in CATIA. Almost every object is associated with a property and method. Property is a characteristic of the object (e.g. get radius size, get hole diameter, get chamfer angle, etc.), while a method is an action performed on the object (e.g. set radius size, set hole diameter, set chamfer angle, etc.) [26]. Most commercial CAD systems store the steps involved in creating a feature, in a construction tree or specification tree as in CATIA.

The *Documents* collection provides access to all files that are loaded in the CATIA session. The *Document* class is an abstract class, meaning it cannot be instantiated to create an object. However, it provides the base functionality that is inherited by all of the specific document types. The *PartDocument* class represents a CATIA part, where geometric modeling is done. Figure 7 shows the object diagram of the *Document* class.

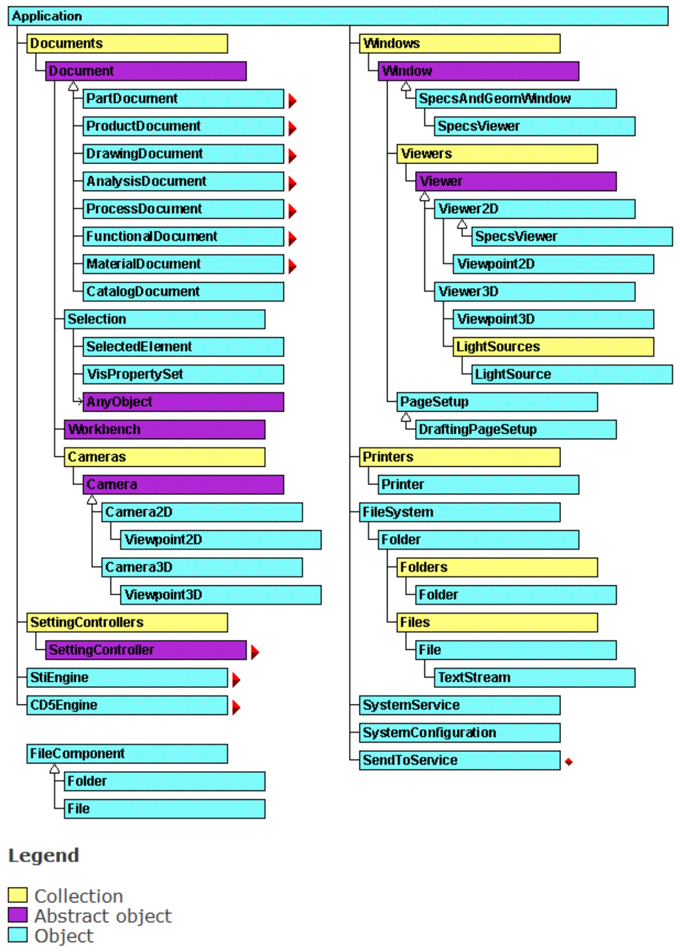


Figure 7: Infrastructure of Automation Objects [27]

The steps performed, from the beginning, to generate a CAD model can be captured in a macro and re-run later to re-create the model without human intervention. Macros can either be scripted manually or be recorded while performing the operations in CATIA. These methods of macro creation are similar to the ones discussed in [28]. Macros allow designers to capture design rule bases and leverage this knowledge to ensure design compliance and to meet standards. Automation of rule bases can be done to incorporate best design practices, design validation etc. [29].

Macros can create either the whole part or individual features on a part. However, in the case of generating individual features, another macro will be required to assemble the features together to create the final part. This method, of using multiple macros to generate features and the part, requires the geometric entities to have unique names. Since macros repeat, every operation recorded in a particular order, errors due to naming of geometric entities, may arise when running multiple macros simultaneously. This is a common issue with automating CAD generation using macros.

2.4 Combination of Methods: Power Copies & Macros with Parameters Table:

The use of macros to instantiate a power copy helps overcome the disadvantages in using these tools separately. Although *VBScript* is powerful in terms of providing access to the user to define basic entities (like points, curves etc.) within CATIA, it is a sequential programming method. Hence, the process of using a macro, in combination with power copy and parameter table, eliminates this disadvantage and provides a modular workflow. Table 1 summarizes the advantages and disadvantages of the methods discussed above.

Table 1: CAD Data Generation Methods

Method	Advantages	Disadvantages
Catalogs + Design Table	- Ease of integration in the CAD model	- Difficult to integrate individual features created using a design table, while satisfying various constraints
Power Copies + Design Table	- Easy to add new features/designs to dataset	- Involves manual work to generate designs by instantiating power copies multiple times
Macros + DOE Table	- CAD model generation is automated	- Sequential process of generating features may lead to inconsistencies (eg. entity naming) - Expanding dataset to include new features is challenging since it requires update to macros
Combination of Power Copies + Macro + DOE Table	- CAD model generation is automated - Easy to expand to include more variations and add new designs/features	- Involve manual work to create power copies (but only once and of the new feature) - Requires renaming certain entities to obtain unique ids - Consistency in naming and arranging geometrical sets is important

In conclusion, the approach of using CATIA macros to instantiate power copies provides a flexible yet powerful workflow to create large dataset for data mining. The method used to generate large datasets is explained in detail, using an example model, in the next section. It is worth mentioning that although this method is currently implemented in CATIA, similar process can be employed in other major CAD software such as Siemens NX or CREO.

3. METHODOLOGY AND CASE STUDY

3.1 Overview

The process of creating large datasets in CAD, using power copies along with DOE tables and macros, requires an initial effort investment in creating power copies for the features involved, but pays off greatly later when generating large number of designs. The proposed methodology is summarized in the flow chart of Figure 8. It starts with the base component, on which various features are created and assembled. Construction entities are created from this base geometry, such as offset planes, wireframe geometry, coordinate systems etc. In the next step, power copies of features, instantiated by a macro, are created. In order to induce variety in designs, there is a need for a set of variables that can be changed without losing the integrity of the geometry. The values of parameters, required to generate each design, are stored in the rows of a table. The structure of this table is similar to a design table in CATIA. Large variety in concept designs are generated by varying the values of parameters, within certain limits, using design of experiments (DOE) methods. The output geometries have to be checked for geometrical integrity. This is partly done in creating each power copy and addressing errors. This is further enforced by applying filtering rules on parameters in the DOE table. The next step is to check the performance of the new concept designs, by setting up the FEA model to run simulations for various load cases.

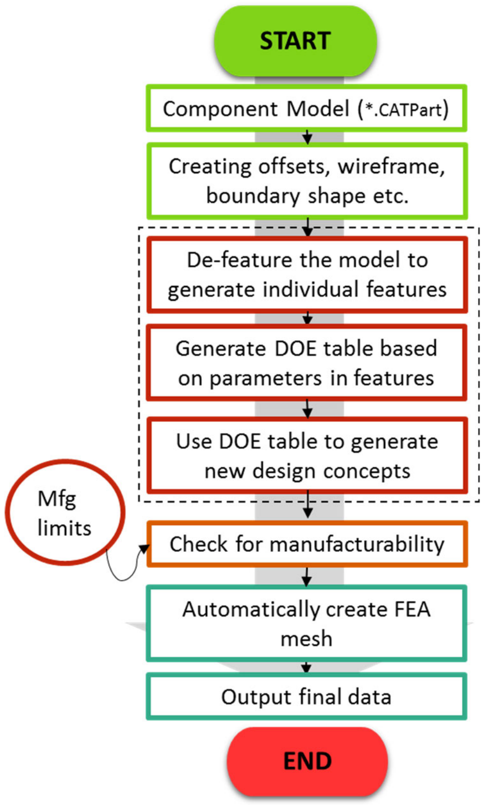

START
Component Model (*.CATPart)
Creating offsets, wireframe, boundary shape etc.
De-feature the model to generate individual features
Generate DOE table based on parameters in features
Use DOE table to generate new design concepts
Mfg limits
Check for manufacturability
Automatically create FEA mesh
Output final data
END

Figure 8: Workflow for Generating Large Datasets of CAD Data

Power copies:

For any component of interest, we need to identify relevant and irrelevant features from existing CAD models. The component is idealized by de-featuring irrelevant features and simplifying the geometry of relevant feature, along with their corresponding parameters. The de-featuring is done such that the overall performance of the component, for critical load cases, is not compromised. In addition, features that do not add structural integrity to the component are eliminated. The critical features are captured in power-copies that are instantiated at a later stage to reproduce the features. The power copy is defined such that the input consists of a single surface on which features are instantiated.

An underlying sketch defines the feature captured in a power copy. The sketch is created using 2D entities like lines, curves etc., on the global planes and then projected onto the corresponding reference surface. Dimensions and constraints applied on the sketch arrest all the degrees of freedom, of the entities, in order to prevent the sketch from breaking when parameter values are updated. The parameters used in the sketch consists of driven and driving parameters. The driving parameters are updated based on the values provided during the creation of the feature, while the driven parameters are updated according to the underling relation or equation.

Figure 9 uses a simple model, to show the use of power copy, for a rib/slot feature. In this figure, the slot feature is created as a power copy, have the following parameters: Slot Length, Width, Height, X Position & Y Position. This power copy can be instantiated on any surface. It can be done automatically on the base surface, which creates the final desired part by executing the series of commands specified in the power copy.

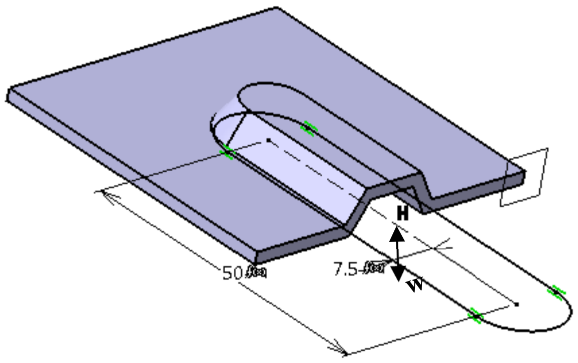
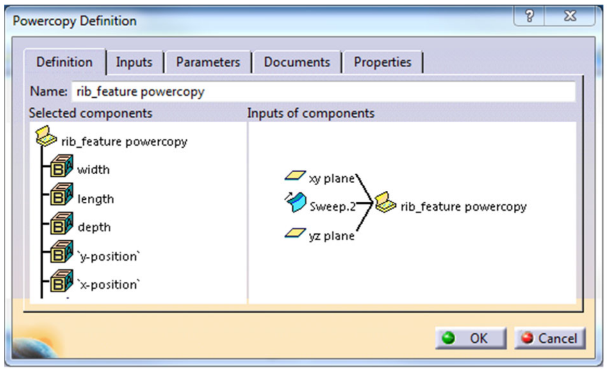


Figure 9: Parameters on the Cylinder Feature

The design table in CATIA drives the parameters that are checked against the constraints applied in the power copy to prevent any geometric violations, before generating the feature.

Once the sketch is created, corresponding surface operations are performed in order to generate the feature. A feature might require multiple surface operations to fully define itself and provide support to a dependent feature, if any. Figure 10 shows the power copy dialoged box for the rib feature along with the required inputs and parameters.



Name: rib_feature powercopy

Selected components
- rib_feature powercopy
  - width
  - length
  - depth
  - `y-position`
  - `x-position`

Inputs of components

Figure 10: Sample Power Copy Creation

Design of Experiments Table:

Parameters like dimension, size, and position of features, defined as an input, are varied within a particular limit in the DOE table. In order to generate variety in designs generated, the DOE table also includes a toggle parameter to switch on or off individual features in a part. Toggle parameters are controlled by 0s and 1s. Combinations of parameter values are generated using DOE methods, to induce variety, while remaining within design boundaries of each feature. Having extreme cases (maximum variation) and a good space filling (intermediate designs) can enhance data mining results; thus, the Box-Behnken method works best as the space-filling algorithm to populate the DOE table. The main difference between a design table in CATIA and DOE table is that the parameter values in the latter, are populated using a DOE method, externally, to ensure the parameter values are set such that it covers the available space within the domain, without intersecting other features. The relation between the performance and parameter combinations/values can be established using a meta-model.

Additionally, at this stage, the interaction between features can be also be controlled by applying additional constraints, known as geometry filtering. The constraints are applied as either lower/upper limits or equations between feature parameters. Such a geometry filter, at an early stage, helps reduce the number of faulty/inconsistent geometries.

Macro:

A CATIA macro instantiates the power copies to generate the features. The macro also assigns the input and parameter values, required by the power copy, from the DOE table one row at a time. The input surface is stored in a separate CATIA part file, on which the power copies are instantiated using the macro. The results are then converted to STEP AP203 files. Figure 11 demonstrates the workflow of the method discussed above. Figure 12 illustrates the pseudo code of the macro to generate CAD models.

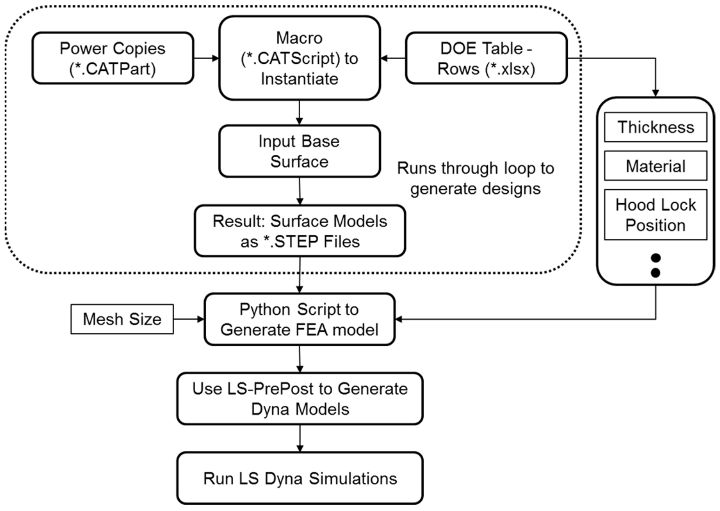

Power Copies (*.CATPart)
Macro (*.CATScript) to Instantiate
DOE Table - Rows (*.xlsx)
Input Base Surface
Runs through loop to generate designs
Result: Surface Models as *.STEP Files
Thickness
Material
Hood Lock Position
Mesh Size
Python Script to Generate FEA model
Use LS-PrePost to Generate Dyna Models
Run LS Dyna Simulations

Figure 11: Flowchart of Hybrid Method

```
initialize loop_variable
loop(loop_variable): Read DOE table
{
        initialize parameter variables
        retrieve current CATIA part
        is_exist(all features)
        {
               if feature == true
                        instantiate feature PC
                                set input objects
                                set new parameters
                else
                        continue
               if feature == true
                        split base surface
                else
                        continue
         }
        join all surfaces
        save *.CATPart and *.stp
 }
```

Figure 12: Pseudo Code for CATIA Macro

After running the macros, all the features selected will be instantiated with different parameter values on different CAD models and depending on the number of variables (n: feature types and m: parameter values) it produces m × n CAD models. These CAD models should be available for analysis to verify the objectives, structural integrity, max deformation, weight, etc. As shown in Figure 13, a Python script generates a command file for LS-PrePost® to read in the STEP files and generate a FEA mesh.

```
initialize loop_variable
loop(loop_variable): read *.stp files
{
       is_exist (OUTPUT_loopvariable.stp)
       change parameters (mesh size etc.)
       generate script_loopvariable.cfile
 }
 initialize variable script_loop
 loop(script_loop): read *.cfile scripts
 {
              is_exist(script_scriptloop.cfile)
              {
                             run_command(script_scriptloop.cfile)
                             save(output_file)
              }
}
run_command(delete *.cfile)
```

Figure 13: Pseudo Code for Mesh Generation

3.2 Case Study

The method/approach discussed in the previous section has enabled the creation of a large dataset of CAD models. This section explains the step-by-step process using a fictitious automotive hood.

Step 1: The base surface of hood skin defined in CATIA serves as the input to generate concept designs. A sample base surface is shown in figure 14.

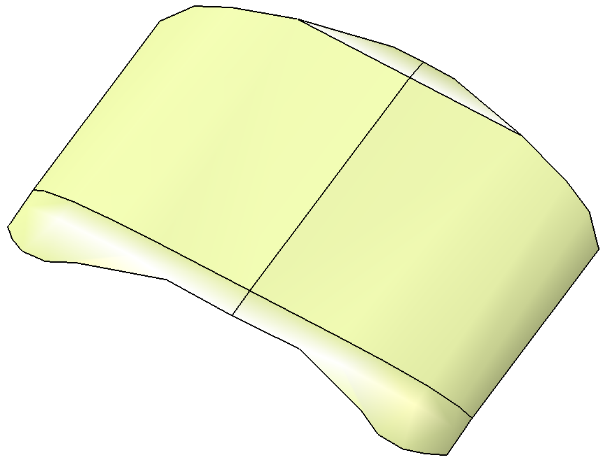


Figure 14: Sample Base Surface of a Hood

Step 2: Features on the hood are identified and captured into power copies with appropriate parameters that are varied in the DOE table. The types of features involved are categorized as: one, features that belong to structural components usually involved in designs of the component of interest, and two, features that involve removal of material to reduce the weight like, pockets and cutout features. The cutout features either are separate power copies or are as part of the power copies for the structural features; the latter being a compound feature. As a compound feature, cutouts and structural features are dimensioned and constrained using the same schema, since the load paths generated by the structural features drive most cutout features. The use of compound features greatly reduces the possibility of unacceptable intersections between features in contrast to the case of defining the features separately. The sketch used to generate one of the structural features is shown in Figure 15.

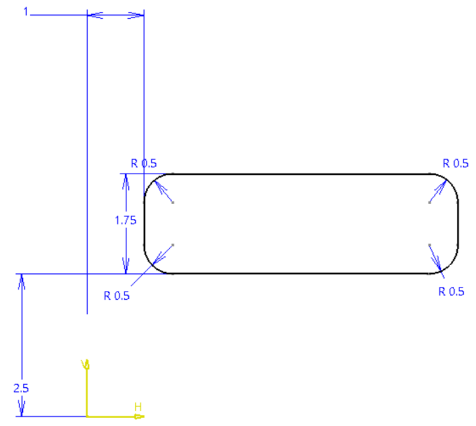


Figure 15: Detailed Sketch to Generate Feature

Figure 16 shows an example of multiple features, stored as power copies, being instantiated by a macro. The design parameters, on these features, are updated from the DOE table each time a power copy is instantiated.

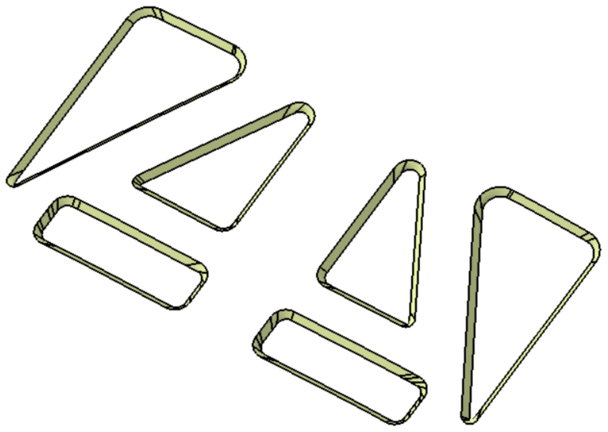
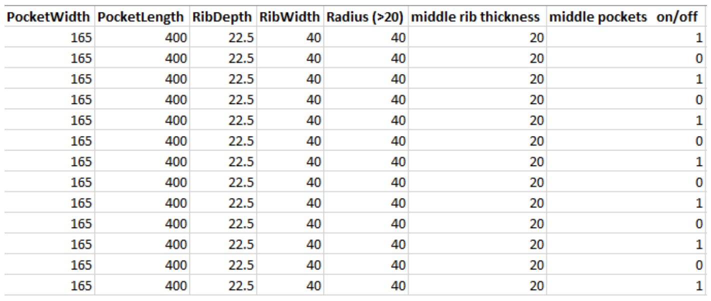


Figure 16: Example of Multiple Features Instantiated by the Macro

Step 3: In this step, a DOE table is created for all the parameters (power copies inputs). In this table feature parameters such as size, location, and depth are varied using DOE methods. In addition, value for toggle parameters are included in the DOE method to create variety. As mentioned in the previous section, each power copy is well dimensioned and constrained. However, when multiple power copies are instantiated using a macro, it could result in undesired intersection of features. Thus, preliminary filtration checks are set in the design table to prevent such geometric violations. For instance, the location of features like ribs, pockets, and lock flats are varied within limits, so that the number of unacceptable geometries, due to undesired intersection, are reduced. Each row in the DOE table corresponds to a unique design. A partial DOE table is illustrated in figure 17.



PocketWidth	PocketLength	RibDepth	RibWidth	Radius (>20)	middle rib thickness	middle pockets on/off
165	400	22.5	40	40	20	1
165	400	22.5	40	40	20	0
165	400	22.5	40	40	20	1
165	400	22.5	40	40	20	0
165	400	22.5	40	40	20	1
165	400	22.5	40	40	20	0
165	400	22.5	40	40	20	1
165	400	22.5	40	40	20	0
165	400	22.5	40	40	20	1
165	400	22.5	40	40	20	0
165	400	22.5	40	40	20	1
165	400	22.5	40	40	20	0
165	400	22.5	40	40	20	1

Figure 17: Partial DOE table

Step 4: In this step, all the power copies, along with the parameters from the DOE table, are instantiated to create the feature of the hood frame, with the base surface as the primary input. A fully instantiated model of a hood, with two different sets of features, hood lock flat, and hinge locations are shown in figure 18.

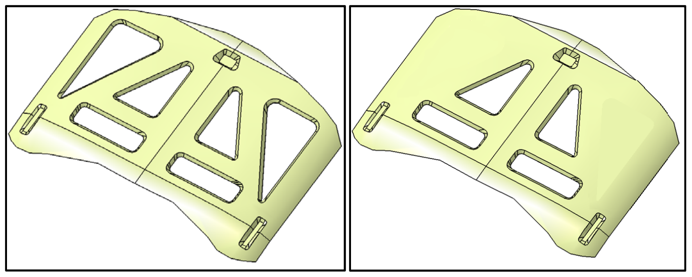


Figure 18: A Simplified Hood Model with Two Sets of Features Instantiated

Step 5: In this case study, the features and parts are created as surface geometry in the Generative Shape Design (GSD) workbench in CATIA. Thus, the result is a surface model with zero thickness. An LS-PrePost® command file is used to set the parameters, like mesh size, and the same is used to generate a mesh. Thickness to the surface mesh is added by *dragging* the elements along the thickness direction and the file is saved in STL format for machine learning. These steps are performed using a python script. The resolution of the STL file is controlled by the size of the elements used during meshing. The element size affects the result in eliminations of features that are smaller than the mesh size. Thus, it is a tradeoff between accuracy of the model representation and storage space required for the model.

The shell mesh can also be setup as a FEA model, with boundary conditions (fixtures and loads), and solved for the various load cases. Setting up the FEA model and solving for results can be done using a python script as well. Figure 19 shows the mesh as an STL file.

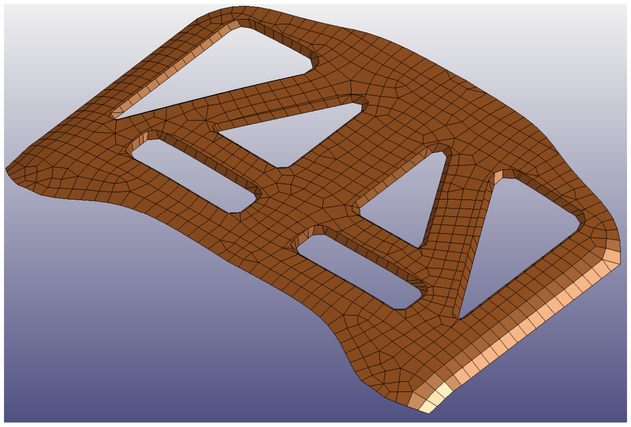


Figure 19: FEA Mesh for STL

4. DISCUSSION & FURTHER RESEARCH

The time taken to generate 100 designs – from CAD to setting up FEA models - takes 30 minutes on a computer with Intel Xeon 2 X 3GHz processor and 32GB RAM. With the use of high performance computing (HPC), the same method can be scaled up to generate our target of over hundred thousand models.

Since the proposed method is feature based CAD model generation, the same attributes can be used for supervised learning of geometries. There are many open source and commercial packages available for deploying neural net (NN) machine learning, such as Matlab Deep Learning Toolbox^{TM} [19] and Google TensorFlow [20]. Future work would include introduction of manufacturing constraints while generating the CAD models, besides the geometry filtering applied in the parameter table. In addition, the next step would also be to compare the results of supervised and un-supervised algorithms that discover features and patterns with deep neural nets based on the performance metrics. A method to automatically setup the FEA model, based on the mesh generated, and perform corresponding simulation needs to be set up.

The concept designss generated, along with the performance metrics, enable the application of novel data mining techniques, in particular, the application of state-of-the-art machine learning methods. For instance, unsupervised deep learning approaches using so-called auto-encoder architectures have been proposed for geometric data by Achlioptas et al. [7] (for geometries represented as unordered point clouds) and Brock et al. [30][8] (for voxel representations). Auto-encoders are a popular approach to learning novel computational representations of the 3D input data [31] and offer the opportunity to identify latent variables underlying the data generation process. These variables may then be used as features in further tasks. Hence, exploring latent representations obtained by data-driven approaches may allow to discover (previously unknown) features or effective combinations of features that result in structures with superior performance. Data-driven approaches may complement and extend current approaches that use pre-defined features based on the expert designer's knowledge on a feature's relevance to performance. Future work should investigate features extracted automatically through unsupervised learning and compare them to pre-defined features, derived from expert knowledge.

In summary, the proposed workflow allows to generate data sets of sizes and structure that enable research on machine learning and artificial intelligence methods for CAD data with the goal to support engineers in the virtual design process. Only the availability of large sets of carefully generated and labeled engineering 3D data allows transfer novel machine learning techniques that have shown unprecedented success in other domains to the automotive design process.

5. ACKNOWLEDGMENTS

The authors would like to acknowledge the financial and technical support from the Honda R&D Ohio, North Americas. The authors would also like to acknowledge the help of Yiming Jiang and Alex Borchers, at the Digital Design & Manufacturing Lab, in generating some of the figures used in the paper.

6. REFERENCES

[1] N. Aulig, E. Nutwell, S. Menzel, and D. Detwiler, "Preference-based Topology Optimization of Body-in-white Structures for Crash and Static Loads," in *14th International LS-DYNA Users Conference*, 2016.

[2] S. Ramnath, E. Nutwell, N. Aulig, and K. Horner, "Detail Design Evaluation of Extruded Sections on BIW," in *15th International LS-DYNA Users Conference*, 2018.

[3] A. Sutradhar, J. Park, J. Kresslein, P. Haghighi, J. J. Shah, and D. Detwiler, "Incorporating Manufacturing Constraints in Topology Optimization Methods: A Survey," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2017, p. V001T02A073.

[4] N. P. van Dijk, K. Maute, M. Langelaar, and F. van Keulen, "Level-set methods for structural topology optimization: a review," *Struct. Multidiscip. Optim.*, vol. 48, no. 3, pp. 437–472, 2013.

[5] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond Euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, 2017.

[6] E. Ahmed *et al.*, "A survey on Deep Learning Advances on Different 3D Data Representations," *Comput. Vis. Pattern Recognit.*, 2018.

[7] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning Representations and Generative Models for 3D Point Clouds," in *35th International Conference on Machine Learning (ICML)*, 2017.

[8] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Generative and Discriminative Voxel Modeling with Convolutional Neural Networks," *arXiv Prepr.*, 2016.

[9] Y. Yu, T. Hur, J. Jung, and I. G. Jang, "Deep learning for determining a near-optimal topological design without

any iteration," *Struct. Multidiscip. Optim.*, vol. 59, no. 3, pp. 787–799, 2019.

[10] N. Aulig, "Generic Topology Optimization Based on Local State Features," Technische Universitaet Darmstadt, VDI Verlag, Germany, 2017.

[11] N. Aulig and M. Olhofer, "State-based representation for structural topology optimization and application to crashworthiness, in Evolutionary Computation," in *2016 IEEE Congress on Evolutionary Computation*, 2016, pp. 1642–1649.

[12] T. Back, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, New York, 1996.

[13] J. Garcke and R. Iza-Teran, "Machine Learning Approaches for Data from Car Crashes and Numerical Car Crash Simulations," in *NAFEMS World Congress 2017: Incorporating the 3rd International Conference on SPDM*, 2017.

[14] H. Zhang, O. Van Kaick, and R. Dyer, "Spectral Mesh Processing," *Comput. Graph. Forum*, vol. 29, no. 6, pp. 1865–1894, 2010.

[15] J. J. Shah and M. Mantyla, *Parametric and Feature-Based CAD/CAM*. John Wiley & Sons, 1995.

[16] A. De Mauro, M. Greco, and M. Grimaldi, "A Formal definition of Big Data based on its essential Features," *Libr. Rev.*, vol. 65, no. 3, pp. 122–135, 2015.

[17] M. Hilbert, "Big Data for Development: A Review of Promises and Challenges," *Dev. Policy Rev.*, 2015.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[19] Mathworks, "MATLAB Deep Learning Toolbox," 2018. [Online]. Available: https://uk.mathworks.com/products/deep-learning.html. [Accessed: 06-Feb-2019].

[20] G. B. Team, "TensorFlow." [Online]. Available: www.tensorflow.org. [Accessed: 06-Feb-2019].

[21] A. Kusiak, "Big Data in Mechanical Engineering." [Online]. Available: https://www.asme.org/career-education/early-career-engineers/me-today/big-data-in-mechanical-engineering. [Accessed: 12-Dec-2018].

[22] L. Wang and C. A. Alexander, "Big Data in Design and Manufacturing Engineering," *Am. J. Eng. Appl. Sci.*, vol. 8, no. 2, pp. 223–232, 2015.

[23] B. Schmitz, "5 Ways Manufacturers will Benefit from 'Big Data,'" 2014. [Online]. Available: https://www.3dcadworld.com/big-data-will-important-manufacturers-future/. [Accessed: 04-Feb-2019].

[24] D. Ha, J. Jongejan, and I. Johnson, "Sketch-RNN Demos." [Online]. Available: https://experiments.withgoogle.com/sketch-rnn-demo.

[25] "About design tables, Catiadoc." [Online]. Available: http://catiadoc.free.fr/online/cfyugkwr_C2/cfyugkwr3005.htm. [Accessed: 04-Feb-2019].

[26] M. Buric and D. Marjanovic, "A Tool for Idealisation of CAD Models," in *International Design Conference - Design 2018*, 2018.

[27] DassaultSystemes, "CAA V5 Visual Basic Help." .

[28] S. Siddesh and B. S. Suresh, "Automation of Generating CAD Models," *J. Mech. Eng. Autom.*, vol. 5, no. 3B, pp. 55–58, 2015.

[29] DassaultSystemes, "CATIA - Knowledge Expert." [Online]. Available: https://www.3ds.com/products-services/catia/products/v5/portfolio/domain/Product_Synthesis/product/KWE/. [Accessed: 06-Feb-2019].

[30] T. Friedrich, N. Aulig, and S. Menzel, "On the Potential and Challenges of Neural Style Transfer for Three-Dimensional Shape Data," in *6th International Conference on Engineering Optimization*, 2018, pp. 581–592.

[31] M. Tschannen, O. Bachem, and M. Lucic, "Recent Advances in Autoencoder-Based Representation Learning," in *Third Workshop on Bayesian Deep Learning*, 2018, pp. 1–26.