ECOLE
Experience-based Computation:
Learning to Optimise

D1.1 – Engineering Data and Descriptors

ECOLE
Experience-based COmputation: Learning to optimisE
766186

H2020 MSCA-ITN
27th March 2020

Authors: Thiago Rios, Stefan Menzel, Bernhard Sendhoff – HRI-EU

1. Introduction 3
2. Geometric Data for Engineering Applications 4
2.1 Geometric Data Manipulation through Morphing Techniques 4
2.1.1 Classification of Car Shapes According to the Projected Frontal Area 5
2.1.2 Classification of CFD Mesh Quality based on FFD Parameters 7
2.2 Engineering Data for Geometric Deep Learning 11
3. Summary and Outlook 14
References 15

1. Introduction

The research in the ECOLE project aims at generating computational models for capturing the notion of experience, which is embedded in data collected over the course of sets of optimizations, and exploiting said experience in similar, yet more challenging, optimization tasks. This vision of experience follows the analogy of an engineer who also collects, abstracts and utilizes her/his professional experience built while working on different types of applications. In the Work Package 1 (WP1) of the ECOLE project, we focus on the research on experience-guided optimization in automotive product design, linking scientific questions to industrial problems and adding a practical perspective to the research tasks.

A central aspect of the research in this domain is defining a target problem and collecting relevant data. Typically, dealing with real-world, automotive application data — here defined as the geometric and engineering performance descriptors associated with a set of different designs — has several obstacles. First, the data available for a specific design or component is usually sparse, since during the development process the design descriptors often change, because they are adapted to local design objectives and constraints, simulation tools and user preferences. Second, evaluating the performance of the design often requires considerable effort, since the results derive from expensive computational simulations, e.g. taking in the range of hours to perform aerodynamics or crash simulations. In addition to the computational cost, the interpretation and documentation of the results can be inconsistent, if e.g. handled by different people. Furthermore, the access to the data might be hindered by the use of file formats that are only compatible with proprietary software tools and by confidentiality agreements.

However, for research purposes, we require reliable and consistent processes to generate large amounts of design data, ideally, online and in real-time. Hence, in order to generate engineering data sets to support ECOLE research activities, we have proposed three general guidelines for designing the algorithms: First, the geometric representations of choice should allow us to modify the designs with a reasonable number of parameters, and to cope with adaptations of the design domain, by adding, subtracting or relocating parameters. Second, the data should be processed in a straightforward and automated fashion, easing the manipulation and transfer of high volume of data. Third, the design modifications and the calculation of the performance metrics should be consistent and the results trackable, in order to ensure reproducibility. This step comprises design pre-processing, simulation and post-processing, e.g. design modification with automated re-meshing, followed by the solution of Navier-Stokes equations to calculate the flow field properties, and automated calculation of the drag force as performance indicator for a car design.

In this report, we present the methods and algorithms developed for the ECOLE project, in order to generate *industrial-grade* data sets for research purposes. The remainder of the report is organized as follows: In Section 2.1, we introduce the morphing technique used to generate synthetic data sets of geometries and corresponding aerodynamic performance. In Section 2.2, we describe the benchmark shape data sets, which provide shapes with higher diversity of geometric features and quality, requiring additional effort in the preprocessing steps. In

Section 2.3, we present a brief overview of work in progress on the automation of geometric data processing to fit benchmark data sets to the requirements of engineering simulations for evaluating their performance. Finally, in Section 3, we conclude the report with a summary and outlook.

2. Geometric Data for Engineering Applications

Following the advances of graphic cards and data storage technology, 3D digital geometric data has become ubiquitous. In order to make sense of the data, engineers use different types of representations for processing geometric data: (a) polygonal meshes, e.g. stereolithography (STL) files, which represent surfaces as undirected graphs, (b) voxels, which encode geometries as their occupancy in a uniform grid, or (c) point clouds, which comprise the coordinates of a set of points sampled from an object [1]. Consequently, the availability of benchmark data sets of shapes has also increased in the past few years, which allowed for the development of more sophisticated tools for geometric processing and powerful deep learning architectures.

However, while these representations encode geometric properties allowing for shape comparisons and analysis, they lack reasonable means to modify the designs with a small number of parameters (user handles). Commercial software tools overcome this issue by using other parameterizations to enable shape manipulation, e.g. Bezier-, B-Splines, and shape morphing techniques, which rely on polynomials to map the geometry to control points, which can be used to generate continuous and intuitive modifications in the parameterized shapes. Therefore, we adopted the morphing approach as a start point for generating data sets of 3D shapes for research within the ECOLE project based on our experience of a series of past design optimization tasks [2-4] and patents in this field [5-7].

2.1. Geometric Data Manipulation through Morphing Techniques

In ECOLE, we apply shape deformation, namely standard free form deformation (FFD) [8], as state-of-the-art method to the generation of shape variations. In this method, the user embeds a geometry in a uniform lattice formed by control points, which are mapped to the geometry through trivariate Bernstein polynomials. Hence, by manipulating the position of the control points, the shape is deformed accordingly and the user can intuitively modify the initial shape, which has its continuity ensured by the order of the polynomials (Figure 1).

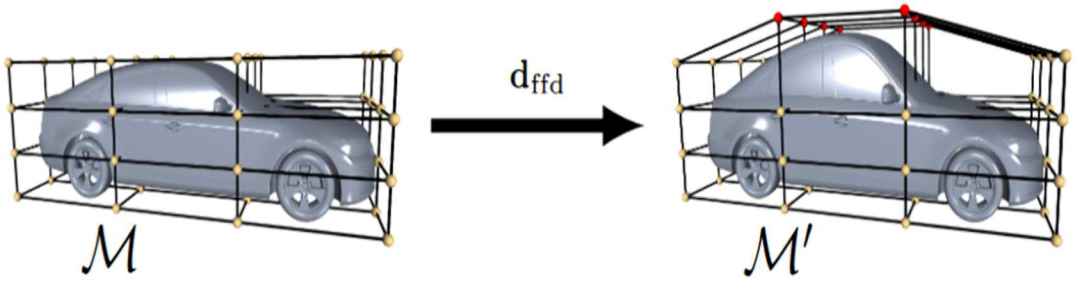

Figure 1 - Example of FFD manipulation applied to the TUM DrivAer model, taken from [9]

The method requires low computational effort and, therefore, it is scalable to high-dimensional computer aided engineering (CAE) models. In addition, it enables the use of different sets of control points to deform the shape and, therefore, allows for an increase in the diversity of modified geometric features and variety of shapes in the data set. In order to generate the data sets, we have implemented a flexible framework, which can be adapted according to the required information. Furthermore, the algorithms were scripted in Python, and supplemented by external files with third-party software specific language whenever required. For reproducing the deformed shapes described in this report, equations (1) and (2) as specified on page 153 of [8] need to be implemented. For each data set, we will provide the set-up coordinates and the deformation parameters used in our experiments.

In the following sections, a description of our research is presented where data sets based on shape deformations of the TUM DrivAer models [10-12] have been employed. In all cases, the input models are STL files with polygonal meshes that represent the TUM DrivAer model. The TUM DrivAer model is a generic car model developed at and provided by the Technical University of Munich allowing researchers and engineers to study aerodynamic effects on a car model. The model includes a set of different car variants and can be accessed via web. (https://www.mw.tum.de/aer/forschungsgruppen/automobilaerodynamik/drivaer/). In the following, we describe the generated data sets along with their purpose within the ECOLE project.

2.1.1. Classification of Car Shapes According to the Projected Frontal Area

The purpose of this data set is to enable studies on resampling techniques for class imbalance research on *industrial-grade* data. The data set designed for the experiments comprises the parameters for 200 deformations of the TUM DrivAer model and their respective estimate of the projected frontal area, as a simplified indicator for aerodynamic performance. In the following, the set-up, namely the FFD and post-processing steps to reproduce the data is described.

The FFD control lattice used to deform the shapes contains three planes in the *x*- and *z*-direction and seven in the *y*-direction, positioned at the coordinates shown in Table 1. In order to select the deformation parameters, we conditioned the deformed shapes to be symmetric with respect to the mid xz-plane, and ensured that control points do not overlap each other, in order to avoid singularities such as self-intersections of the design surfaces. To enforce the symmetry of the models, the position of the control points was mirrored with respect to the mid *xz*-plane of the lattice. Hence, considering the proposed constraints, we defined the axial displacement of the control planes as deformation parameters, which results in nine design variables, as shown in Figure 2.

Table 1 – Position of the control planes along each axis.

Direction	Vehicle Boundaries [m]	Position of the planes [m]
x	[-0.88, 3.80]	[-0.88, 2.34, 3.80]
y	[-1.02, 1.02]	[-1.02, -1.36, -0.68, 0, 0.68, 1.36, 2.04]
z	[-0.31, 1.09]	[-0.31, 1.76, 3.20]

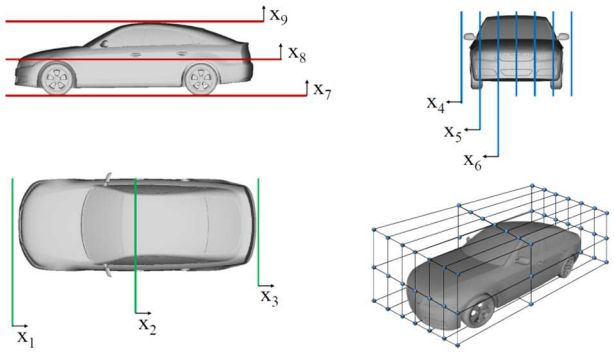

Figure 2 – Control parameters and lattice used to deform the DrivAer model

One challenge during an automated geometry generation process is to ensure that all deformed designs are valid, i.e. free of self-intersections of the design surface. Invalid designs would fail during a potential CFD simulation. Therefore, we set constraints on the maximum movement of each control point to avoid a change in the order of control points which is a simple heuristic for valid designs. We used a uniform distribution to randomly sample from the design space. To ensure reproducibility we started the random number generator with a defined *seed* value.

To assign a performance value to each car design, we estimated the projected frontal area by the sum of the projection of the area of the STL elements to the frontal plane of the car, according to the element normal direction. Although it is a low fidelity approximation of the aerodynamic performance, calculating the projected area requires much less computational effort than CFD simulations, which enabled us to generate enough volume of data within a reasonable amount of time.

Finally, the workflow of the data for the complete process is shown in Figure 3. As previously mentioned, the deformed models derive from the same initial shape, which is embedded in the lattice. Based on the number of planes and their positions along the axes, the deformation parameters are sampled and stored in a temporary text file. Then, until the number of required deformations is generated, the algorithm performs a loop, where the parameters are assigned to the lattice, the shape is deformed, the projected area is calculated and the data-set of parameters and projected area are stored in a dat file, structured as shown in Table 2.

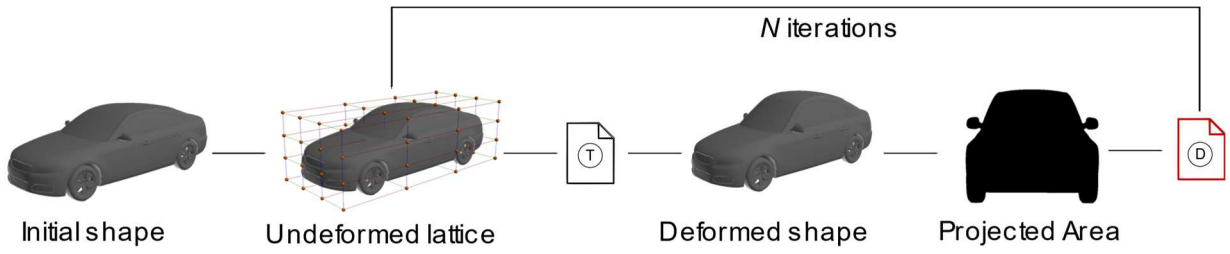

Figure 3 – Workflow of the data generation process

Table 2 - Structure of the text file containing the data set

File name: D1_TUMDrivAerFrontalArea-FFDParametersAndMetrics-190301-RESTRICTED.dat

ID	x1	x2	x3	x4	x5	x6	x7	x8	x9	Area
DEFORM_1	-0.927	1.823	4.154	-1.133	-0.818	-0.365	-0.515	0.494	1.550	4.702
…	…	…	…	…	…	…	…	…	…	…
DEFORM_200	-1.354	1.459	4.467	-0.727	-0.740	-0.451	-0.501	0.325	1.132	5.905

According to the ECOLE data management plan document (D5.3), the data is classified as restricted, i.e., the data set can be accessed via the Bear Data sharing repository of the University of Birmingham (https://beardatashare.bham.ac.uk/login) after registration has been requested and completed, or by contact through the form provided on the ECOLE webpage: https://ecole-itn.eu/contact/. Although we have gone through some effort to document the data, its use is not entirely self-explanatory. The complexity of the engineering process is simply too high and the effort would be unreasonable to make the data set completely self-contained. Therefore, the required registration also allows us to provide some support and additional explanation to interested researchers.

2.1.2. Classification of CFD Mesh Quality based on FFD Parameters

The purpose of this data set to allow the analysis of the effects of resampling techniques on class imbalance classification problems for distinguishing between valid (correctly converged) and erroneous CFD results. As a first criteria, we observe the quality of the deformed CFD mesh, which indicates the expected quality of the CFD calculations. A low quality CFD mesh with many stretched grid cells or even intersecting grid cells does not converge to reasonable aerodynamic results. Hence, we selected the prediction of mesh quality as an industrial application, where a classification algorithm is used to distinguish sets of FFD parameters into two groups: the first group would result in feasible CFD mesh deformations which guarantee valid CFD simulations while the second group would result in ill-defined CFD meshes which would result in erroneous CFD simulation results. Also differently from the previous case, this study requires the FFD algorithm to operate directly on the CFD mesh (Figure 4), as it is usually done in the actual optimization process [2-4]. Hence, the previous process (Section 2.1.1) to generate the data was modified in such a way that the scripts would interact with a third-party software, namely the CFD simulation tool OpenFOAM [13]. Furthermore, restrictions imposed by the CFD simulations had to be taken into account. Finally, the software had to run on a high-performance computer cluster.

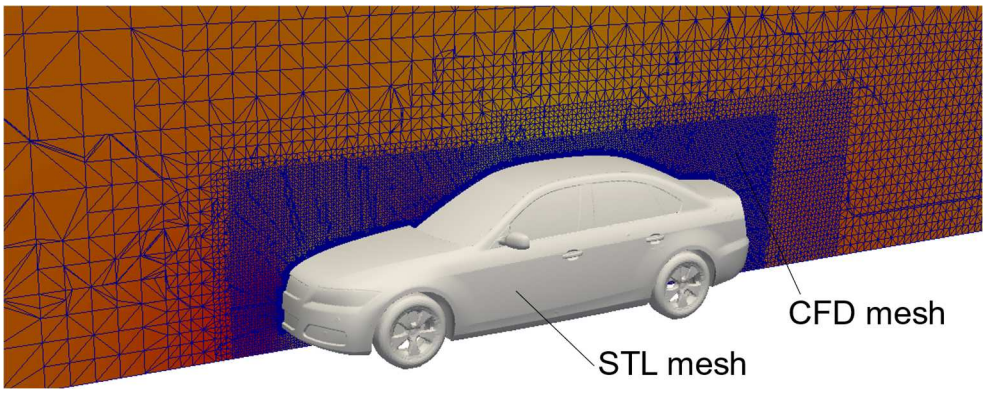

Figure 4 – Representation of the STL mesh, used to describe the car shape, and the CFD mesh, which is used for calculating the air flow around the car.

For the CFD simulations, it is assumed that a symmetric car shape is positioned in a large fluid domain, with the wheels tangent to the ground and rotating with speed compatible to the linear velocity of the vehicle, which rides in a straight line and aligned with the flow. In order to keep the assumptions valid for the simulation of any deformed car shape, FFD should preserve the position of the car as much as possible, and thus, we defined a different lattice to the one described in Section 2.1.1 to perform the deformations.

The lattice used in the experiments had an offset distance from the extremes of the initial shape, except on the bottom, where it was kept tangent to the tires (Table 3). In order to preserve the position of the car, we fixed two control planes adjacent to each boundary of the lattice. Additionally, this constraint also attenuates the distortion of elements close to the limits of the lattice, which do not affect the shape of the car but are critical to the quality and convergence of the simulation results. Hence, the final control volume comprises of seven planes in the *x*- and *z*-direction and ten planes in the *y*-direction, as shown in Figure 5.

Table 3 – Boundary values measured from the car shape and used to define the lattice.

Direction	Vehicle Boundaries [m]	Lattice Boundaries [m]
x	[-0.88, 3.80]	[-1.50, 5.00]
y	[-1.02, 1.02]	[-2.00, 2.00]
z	[-0.31, 1.09]	[-0.31, 3.20]

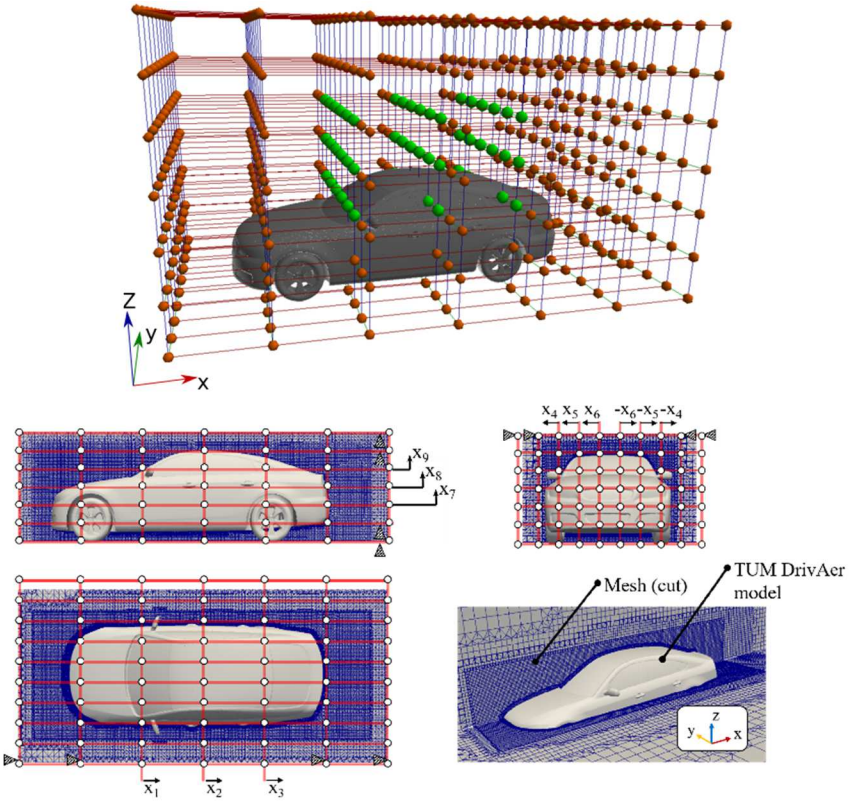

Figure 5 – Control volume used to generate the data set.

The parameterization of the model is very similar to the case we presented previously. We selected the axial displacement of the remaining sections of the control planes. Considering the constraints and imposing symmetry of the lattice with respect to the mid *xz*-plane, it resulted in 9 deformation parameters. The values of the parameters were randomly sampled according to a uniform distribution, and on intervals defined such that any pair of control planes would overlap (Figure 6), as in the previous case to avoid invalid designs.

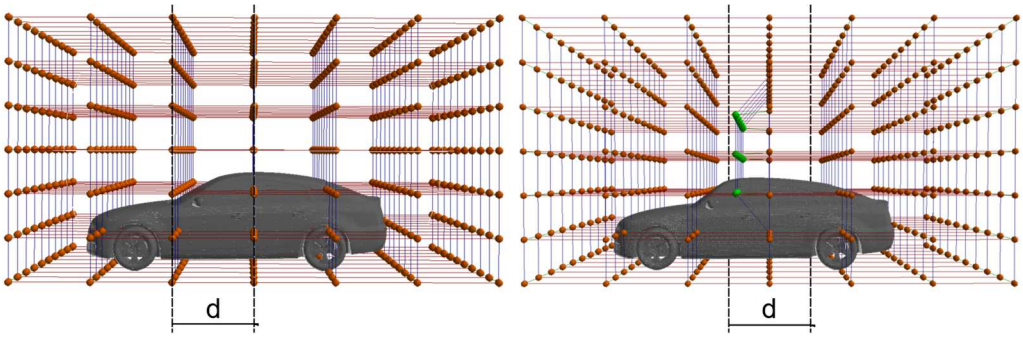

Figure 6 – Example of interval for moving the control points in the x-direction.

The process to generate the data set is comprised of three main steps: (a) generating the directories with configuration files following the OpenFOAM requirements; (b) running the meshing, mesh quality assessment and FFD algorithms; (c) running the CFD simulations and (d) post-processing the results. The shape deformation is fundamentally the same as performed for

the previous data set, however the parallelization of the data processing and the use of third-party software requires a more sophisticated workflow.

In the first step, or pre-processing step, the objectives are to determine the structure of the FFD lattice, to sample the deformation parameters and to prepare the data structure to ensure compatibility with OpenFOAM. The parameterization and sampling were performed as in the previous case (scripted in a Python algorithm) and the algorithm resulted in two outputs: a text file with the parameter values and a directory for each case of the deformation (Figure 7), following the OpenFOAM file structure.

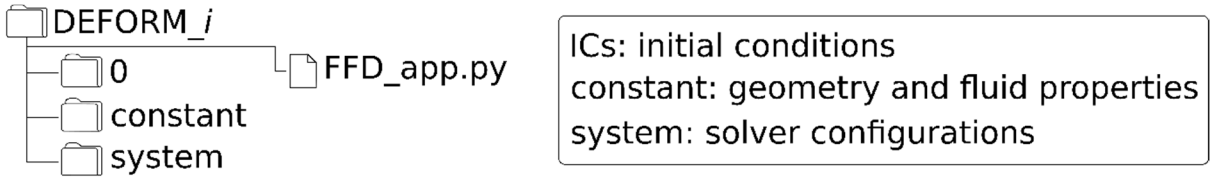

Figure 7 – Structure of the directory used by OpenFOAM to perform the simulations

After pre-processing, the next step is comprised of the generation, deformation and quality assessment of the meshes. The OpenFOAM algorithms *blockMesh*, *snappyHexMesh* and *checkMesh* generated the mesh on the fluid domain based on the initial shape, which was the same for all the cases. The initial mesh quality was evaluated based on the number of warnings, the maximum skewness and the extreme values of the aspect ratio [14]. In order to speed up the data generation, the algorithms were parallelized on 16 processors for each case, decomposing the mesh volume into 16 parts that were stored on individual subdirectories, as shown in Figure 8. The metrics, warnings and further information yielded by the algorithms were stored in log files, which could be later post-processed.

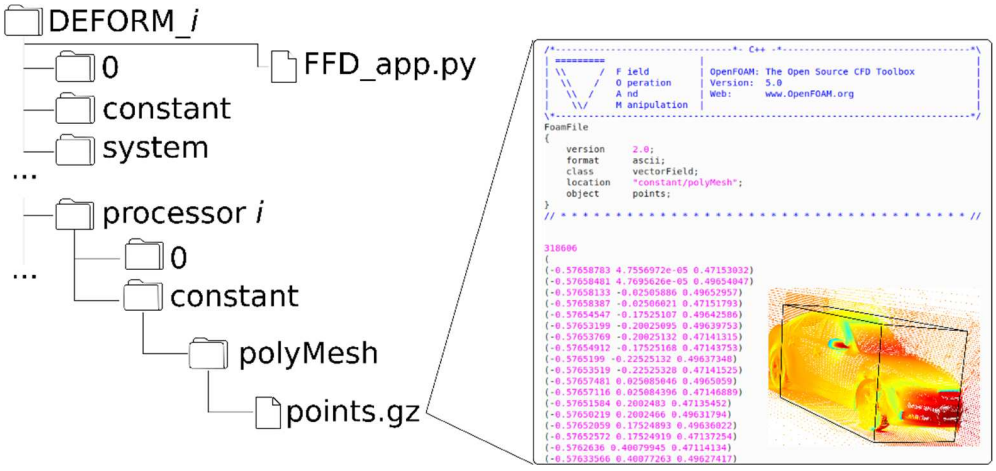

Figure 8 – Structure of the simulation directory after meshing the initial domain. The directory of each processor contains the solutions and mesh information related to 1/n-th of the domain, where n *is the number of processors.*

When the initial meshing was finished, for each case, a Python script with the implementation of the FFD algorithm was called with the deformation parameters as arguments. In order to deform

the mesh, the script accessed the files with the coordinates of the mesh nodes of each partition of the domain and applied the FFD algorithm on the nodes using the same initial lattice and deformation parameters. Here, a large portion of the domain was not embedded into the lattice, therefore, a conditional statement was added to the algorithm, in order to filter the points read from the files and to ensure that the FFD operation was applied only on the points within the lattice span. In addition, the OpenFOAM code was written in C++ language, such that the files with the mesh properties had a specific layout that is required by the language. Hence, the Python script was complemented with a text manipulation section prior and after the FFD operations. This ensured that the reading and writing operations followed the OpenFOAM templates. After deforming the model, the *checkMesh* algorithm performed the quality assessment of the deformed mesh, appending the metrics to the initial log file, and the fluid flow is calculated for the deformed meshed.

After all simulations finished, the post-processing step was performed. Here, a Python algorithm reads the *checkMesh* log files, and restores the quality metrics of the deformed meshes. After that, the same algorithm retrieves the forces acting on the car shape from the simulation output files and derives the average drag and lift forces over the last 50 steps of the simulation. Finally, the obtained values were appended to the file with the deformation parameters, storing the complete set of information in a final text file, which is structured as shown in Table 4. In total, we generated 300 mesh deformations. Additional details, including the application of the data set, are documented in [15].

Table 4 - Structure of the data set containing the mesh quality measurements

File name: D2_TUMDrivAerMeshQuality-FFDParametersAndMetrics-191101-RESTRICTED.dat

ID	x_1	x_2	…	x_9	check_failures	max_skewness	max_aspect_ratio	F_x	F_z
DEFORM_1	0.076	0.336	…	0.349	5	1.58E+02	3.41E+97	408.471	671.012
…	…	…	…	…	…	…	…	…	…
DEFORM_300	0.765	0.626	…	-0.055	5	528.441	6.97E+98	587.370	41.657

According to the ECOLE data management plan document (D5.3), the data is classified as restricted, i.e., the data set can be accessed via the Bear Data sharing repository of the University of Birmingham (https://beardatashare.bham.ac.uk/login) after registration has been requested and completed, or by contact through the form provided on the ECOLE webpage: https://ecole-itn.eu/contact/.

2.2. Engineering Data for Geometric Deep Learning

The main advantages of working on synthetic data sets are the control over the number of samples and the distribution of features. Such characteristics enable us to test and validate new machine learning and optimization methods in a controlled environment (e.g. adjustable noise) improving on the current state of the art based on the underlying theory. However, in real-world applications data is often noisy (with non-trivial distributions) with different types of parameterization and the distribution of features might also be unknown. Therefore, in order to test our algorithms on an environment closer to real-world conditions, we use benchmark

geometric data sets. In addition, public benchmark geometric data sets allow different research groups to evaluate novel methods on standard geometries and compare results between different approaches.

Geometric deep learning is a novel field in geometry processing where deep neural networks are utilized to find compact representations based on unsupervised learning. In literature, the models used for geometric deep learning often derive from the ShapeNet [16], ModelNet [17] or ABC [18] data sets. The first and second data sets contain about 52.000 and 128.000 labeled 3D models, respectively, which can be employed for data compression and classifications tasks [19-21]. The ABC data set comprises one million 3D shapes, mostly mechanical components, that have been employed for the research for determining surface normal angles [18] and may be used for further applications, such as 3D-edge detection and shape reconstruction algorithms [22-24]. For the ECOLE research activities, so far, we have used the models from ShapeNet [16] for data compression with point cloud autoencoders. Therefore, we focus in this report on the characteristics of this specific data set.

ShapeNet is a repository of 3D shapes from multiple semantic classes, categorized according to the Wordnet synsets [25]. Many of the models in the data set also contain annotations regarding names, geometric and physical properties, which can be used in classification and regression tasks. The repository can also be partitioned in subdivisions with cleaner models, i.e. single objects with the same orientation and manually verified, such as ShapeNetCore, or more densely annotated geometries. These models were used for testing different geometric deep learning architectures [21, 26-28]. In the context of the ECOLE project, we are particularly interested in point cloud autoencoders, focusing on data compression as a potential method for increasing the efficiency of high-dimensional CAE models.

Hence, for training and testing point cloud autoencoders, we used the shapes in the car class of ShapeNetCore. More specifically, we used about 7500 point clouds with 2048 points uniformly sampled from the shapes, as used in [21] and made available by the authors. In the data set, the models are organized in directories according to the classes identified by an 8-digit numeric code. In the directories, the point clouds are stored as Polygon File Format (PLY) files and identified by unique alpha-numeric code with 32 digits. The models can be directly manipulated using third-party software, such as Meshlab, however the authors in [21] provided Python scripts to decode the classes identifications into their names and retrieve the point clouds for training the deep learning models. The architectures validated using the ShapeNetCore data set were applied on studies published at the IEEE Symposium Series on Computational Intelligence (SSCI) in 2019 [29, 30].

However, while datasets like ShapeNet or ShapeNetCore contain the geometries of various objects, they do not provide engineering metrics like e.g. aerodynamic performance of different car models. Therefore, within the ECOLE project we take sets of geometries from these databases, eventually apply preprocessing techniques such as resampling or mesh cleaning and afterwards simulate them using OpenFOAM for aerodynamic performance.

Aligned with this objective, we are testing a process for preparing the original models from ShapeNetCore, saved as OBJ files, to be used in CFD simulations and geometric deep learning with higher-dimensional models (Algorithm 1). The process consists of Python and Meshlab server scripts, which scale and position the models with respect to the assumed setup for the CFD calculations, as well as convert the file format to STL and XYZ, which already have been used in previous tasks. Furthermore, directories with the CFD configuration files for OpenFOAM are generated, based on a template, such that the folders can be transferred directly to the simulation environment.

Algorithm 1: Data generation from ShapeNetCore car class

INPUT: Shapes of the car class from ShapeNetCore
Target refinement level (number of iterative element subdivisions, minimum edge length)
Target directories to store the refined STL and point cloud models
OpenFOAM simulation templates
Required number of models

FOR each model **IN** the ShapeNetCore data set:

 # Load the data

1. Convert from the original format (OBJ) to STL, preserving the mesh characteristics

2. Import the mesh to the Python environment as a variable and extract the nodes

 # Geometric operations

3. Rescale the nodes coordinates from $[-1, 1]^3$ to real size dimensions

4. Align and position the model based on the setup of the CFD simulation

5. Refine the model using Meshlab server, following the criteria given as input

 # Calculate geometric properties

6. Calculate geometric features of interest, for example, surface area and volume, and store the calculated values

 # Save models and features

7. Create directory with unique name (ID), based on the openFOAM template, and save the refined STL model in the corresponding location

8. Convert the saved STL to a point cloud and save it in the target directory (input) as a XYZ file. The conversion and removal of point duplicates is performed with meshlab server

9. Save the calculated features in a temporary text file, where the first column should contain the unique ID of the model

IF the required number of models was achieved: **BREAK**

Save the name and corresponding features of the models in a final text file

---

For preliminary studies we generated a data set with the simulation results of drag and lift forces for 400 shapes from ShapeNetCore, structured as shown in Table 5. The data comprises an identifier to find the corresponding shape within the ShapeNetCore dataset along with the drag and downforce value calculated using our preprocessing process (Algorithm 1) and OpenFOAM.

Table 5 – Structure of the data set with the results of aerodynamic simulation on ShapeNetCore car models

File name:
D3_CFDExperimentsOnShapeNetCore_CFDForceResults_20200228_RESTRICTED.dat

ID	F_x	F_z
shape_51b8011a2eaaed85cde8c99e4b182f9	408.471	671.012
…	…	…
shape_6283a8fcea4976fe47bff85f09fd66b	587.370	41.657

According to the ECOLE data management plan document (D5.3), the data is classified as restricted, i.e., the data set can be accessed via the Bear Data sharing repository of the University of Birmingham (https://beardatashare.bham.ac.uk/login) after registration has been requested and completed, or by contact through the form provided on the ECOLE webpage: https://ecole-itn.eu/contact/. Since we expect that these aerodynamic performance values will be of interest to a larger group of researchers, we aim to improve the dataset to be closer to actual engineering cases and make it directly publicly available without registration. The target is to achieve stable aerodynamic performance values for a larger set of geometries of the ShapeNetCore data set. We plan to announce the publication of our data set via the ECOLE webpage and twitter account.

3. Summary and outlook

Collecting results from multiple optimization runs and design cases is central to building a base from which machine learning models can abstract design features and, ideally, learn the embedded experience of the expert. Therefore, we developed different data generation algorithms and simulation processes to create data sets for training and testing machine learning algorithms in the framework of the ECOLE project.

In this report, we have presented the processes to manipulate and perform simulations on CAE models, which yield structured data sets for machine learning purposes. Our methods rely on publicly available benchmark shapes, such as the TUM DrivAer model, and free form deformation, a state-of-the-art shape morphing technique. In the discussion about the methods, we also presented the parameters used to generate the cases contained in the data sets, such that the results can be reproduced in future experiments. Furthermore, the data comprise not only geometric information, but also engineering metrics, such as mesh quality and aerodynamic forces, such that they can be employed in different machine learning tasks, such as geometric deep learning and clustering.

Finally, the generation of data sets is a work in progress. Due to the uniqueness of multiple topics covered by the ECOLE project, the processes are tailored for each application. Therefore, the work presented in this report will be extended in the future, aggregating more engineering performance metrics and different types of shapes.

In this report, we have provided the necessary information to allow the reproducibility of the data sets that we have generated. The equations used to implement the FFD algorithm are

available in [5] and can be scripted in different programming languages, depending on the final application. By assigning the same control lattice configurations and range of deformation parameters, an informed user should be able to reproduce the results contained in the files described in this report. The files containing the original geometries of the TUM DrivAer model and ShapeNetCore must be requested directly from their respective developers. Extensive descriptions of the TUM DrivAer model and access to the geometry files can be found at https://www.mw.tum.de/aer/forschungsgruppen/automobilaerodynamik/drivaer/ (accessed on 04. March 2020). ShaepeNetCore geometries and descriptions can be found at https://www.shapenet.org/ (accessed on 04. March 2020).

According to the ECOLE data management plan document (D5.3), all three datasets D1, D2 and D3 are classified as restricted, i.e., the data set can be accessed via the Bear Data sharing repository of the University of Birmingham (https://beardatashare.bham.ac.uk/login) after registration has been requested and completed, or by contact through the form provided on the ECOLE webpage: https://ecole-itn.eu/contact/.

References

[1] T. Friedrich, N. Aulig, and S. Menzel, “On the potential and challenges of neural style transfer for three-dimensional shape data”, in International Conference on Engineering Optimization. Springer International Publishing, 2019, pp. 581–592.
[2] S. Menzel, M. Olhofer, and B. Sendhoff, “Application of free form deformation techniques in evolutionary design optimisation”, in 6th World Congress on Structural and Multidisciplinary Optimization (WCSMO6), J. Herskovits, S. Mazorche, and A. Canelas, Eds. Rio de Janeiro: COPPE Publication, 2005
[3] S. Menzel and B. Sendhoff, “Representing the Change - Free Form Deformation for Evolutionary Design Optimization”, in Evolutionary Computation in Practice, Tina Yu and David Davis and Cem Baydar and Rajkumar Roy (eds.), Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 63–86. [Online]. Available: https://doi.org/10.1007/978-3-540-75771-9_4
[4] M. Olhofer, T. Bihrer, S. Menzel, M. Fischer, and B. Sendhoff, “Evolutionary optimisation of an exhaust flow element with free form deformation”, in Simulation for Innovative Design, Proceedings of the 4th EASC - 2009 European Automotive Simulation Conference, K. Seibert and M. Jirka, Eds. ANSYS Inc., 2009, pp. 163–174.
[5] S. Menzel, M. Olhofer, B. Sendhoff, “Evolutionary optimization and free form deformation ”, US7609262B2, granted, 2009.
[6] S. Menzel, M. Olhofer, B. Sendhoff, “Evolutionary design optimization using extended direct manipulation of free form deformations”, US Patent US 7982732B2 , granted, 2011
[7] S. Menzel, M. Olhofer, B. Sendhoff, “Evolutionary direct manipulation of free form deformation representations for design optimization”, US Patent US8098244B2, granted, 2012.
[8] T. W. Sederberg and S. R. Parry, “Free-form deformation of solid geometric models”, in Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, ser. SIGGRAPH ’86. New York, NY, USA: ACM, 1986, pp. 151–160. [Online]. Available: http://doi.acm.org/10.1145/15922.15903

[9] S. Saha, T. Rios, L. Minku, X. Yao, Z. Xu, B. Sendhoff, and S. Menzel, “Optimal Evolutionary Optimization Hyperparameters to Mimic Human User Behaviour”, in 2019 IEEE Symposium Series on Computational Intelligence (SSCI), Xiamen, China, 6-9 December 2019.

[10] A. Heft, T. Indinger, N. Adams: Experimental and Numerical Investigation of the DrivAer Model, ASME 2012, July 8-12, 2012, Puerto Rico, USA, FEDSM2012-72272

[11] A. Heft, T. Indinger, N. Adams: Introduction of a New Realistic Generic Car Model for Aerodynamic Investigations, SAE 2012 World Congress, April 23-26, 2012, Detroit, Michigan, USA, Paper 2012-01-0168

[12] A. I. Heft, T. Indinger, N. A. Adams: Investigation of Unsteady Flow Structures in the Wake of a Realistic Generic Car Model, 29th AIAA Applied Aerodynamics Conference, June 27-30, 2011, Honolulu, Hawaii, USA, Paper AIAA 2011-3669

[13] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby, “A tensorial approach to computational continuum mechanics using object-oriented techniques”, Computers in Physics, vol. 12, no. 6, pp. 620–631, 1998. [Online].Available: https://aip.scitation.org/doi/abs/10.1063/1.168744

[14] P. Knupp, “Measurement and impact of mesh quality (invited)”, 46th AIAA Aerospace Sciences Meeting and Exhibit. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2008-933

[15] Jiawen Kong, Thiago Rios, Wojtek Kowalczyk, Stefan Menzel and Thomas Bäck, “On the Performance of Oversampling Techniques for Class Imbalance Problems” in the 24th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Singapore, 11-14 May 2020 (Accepted).

[16] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, “ShapeNet: An information-rich 3D model repository”, arXiv preprint arXiv:1512.03012v1 [cs.GR], 2015.

[17] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao 3D ShapeNets: A Deep Representation for Volumetric Shapes Proceedings of 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR2015)

[18] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo, “ABC: A big CAD modeldataset for geometric deep learning”, CoRR, vol. abs/1812.06216, 2018.[Online]. Available: http://arxiv.org/abs/1812.06216

[19] Y. Liu, B. Fan, S. Xiang, and C. Pan, “Relation-shape convolutional neural network for point cloud analysis”, CoRR, vol. abs/1904.07601,2019. [Online]. Available: http://arxiv.org/abs/1904.07601

[20] Z. Zhang, H. Lin, X. Zhao, R. Ji and Y. Gao, "Inductive Multi-Hypergraph Learning and Its Application on View-Based 3D Object Classification”, in IEEE Transactions on Image Processing, vol. 27, no. 12, pp. 5957-5968, Dec. 2018.

[21] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, “Learning representations and generative models for 3D point clouds”, in Proceedings of the 35th International Conference on Machine Learning (ICML), vol. 80., Stockholm Sweden: PMLR, 2018, pp. 40–49.

[22] C. Weber, S. Hahmann, and H. Hagen. Sharp feature detection in point clouds. In Proceedings of the 2010 Shape Modeling International Conference, SMI ’10, pages 175–186, Washington, DC, USA, 2010. IEEE Computer Society.

[23] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng. Punet: Point cloud upsampling network. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[24] X. Sun, P. L. Rosin, R. R. Martin, and F. C. Langbein. Noise analysis and synthesis for 3d laser depth scanners. Graphical Models, 71(2):34 – 48, 2009. IEEE International Conference on Shape Modeling and Applications 2008.

[25] George A. Miller. WordNet: a lexical database for English. CACM, 1995. 1, 2, 3, 4.

[26] Y. Yao, N. Schertler, E. Rosales, H. Rhodin, L. Sigal, and A. Sheffer, "Front2back: Single view 3d shape reconstruction via front to backprediction", 2019.

[27] Q. Lu, Y. Lu, M. Xiao, X. Yuan, and W. Jia, "3d-fhnet: Three-dimensionalfusion hierarchical reconstruction method for any number of views", IEEEAccess, vol. 7, pp. 172 902–172 912, 2019.

[28] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "Cooperative object classification for driving applications", 2019 IEEE Intelligent Vehicles Symposium (IV), pp. 2484–2489, 2019.

[29] T. Rios, T. Bäck, B. van Stein, B. Sendhoff, and S. Menzel, "On the efficiency of a point cloud autoencoder as a geometric representation for shape optimization", in 2019 IEEE Symposium Series on Computational Intelligence (SSCI), Xiamen, China, 6-9 December 2019.

[30] T. Rios, P. Wollstadt, B. van Stein, T. Bäck, Z. Xu, B. Sendhoff, and S. Menzel, "Scalability of Learning Tasks on 3D CAE Models using Point Cloud Autoencoders", in 2019 IEEE Symposium Series on Computational Intelligence (SSCI), Xiamen, China, 6-9 December 2019.