NordDesign 2020
August 11-14, 2020
Kgs. Lyngby, Denmark

Comparison of Design Automation and Machine Learning algorithms for creation of easily modifiable splines

Erik Anton Gustafsson[1], Johan Alexander Persson[1], Johan Rolf Ölvander[1]

[1] Linköping University
erik.gustafsson@liu.se
johan.persson@liu.se
johan.olvander@liu.se

Abstract

In order to enable easy modification of results from a design optimization process in a CAD tool, a flexible representation of the geometry is needed. This is not always trivial however, since many file formats are not importable as modifiable geometry into the CAD tool, and if they are, they might not represent the geometry in a way that enables easy modification. To mitigate this problem a design automation (DA) and a machine learning (ML) approach are developed and compared using a test case from an optimization process used to optimize hose routing in tight spaces. In the test case used, the geometry from the optimization process consists of center curves represented as a large number of points. To enable easy modification a more flexible representation is needed such as a spline with a few well-placed control points. Both the DA and ML approach can approximate center curves from the optimization process as splines containing a varying number of control points but do show different properties. The DA approach is considerably slower than the ML but adds a lot of flexibility regarding accuracy and the number of control points used.

Keywords: Design Automation, Machine Learning, Computer aided Design, Optimization

1 Introduction

In order to stay competitive on a global market companies strive towards the development of "better" products and a more efficient product development process. One means to meet these objectives is to automate parts of the development process by introducing techniques such as design automation, formal optimization and AI inspired techniques such as machine learning.
In the case studied in for this paper, further described in (Wehlin et al., 2020) a multi-objective optimization (MOO) has been set up for hose routing assemblies containing several hoses that all share a common space, a complex task that requires a lot manual engineering effort. Although the solutions take several disciplines into account, they still need adjustments and refinements from an engineer before a final design is achieved. For this reason, it is desirable to generate solutions that are easily modified by the engineers after the optimization has been conducted. When it comes to problems where the geometric shape of the product is of

importance, the incorporation of different software in the optimization framework can limit the possibility to easily manipulate the geometry since different software use different representations of the geometry. This means that the geometry needs to be recreated manually if the data format from the optimization cannot be imported as modifiable geometry to the CAD software used when creating the final design.

The MOO process studied in this work is represented in figure 1. First the problem is defined in the CAD software, CATIA v5 in this framework, where the surrounding geometry and start and end nodes for each hose are entered. An Excel interface is then used to define settings for the optimization and to communicate with the routing software which is Industrial Path Solutions (IPS)[1] in this case. IPS calculates the routes based on input from the optimization algorithm and creates simulated hoses for each configuration, an example of this can be seen in figure 2. When the optimization is completed, evaluation and fine-tuning is again done in the CAD software, meaning that the result from IPS must be imported to the CAD tool. The exported configurations from IPS consists of a Virtual Reality Modeling Language (wrl) file for each hose in the configuration, each containing a series of points representing the center curve for a single hose. In the CAD software however, it is desirable to have as few points as possible when recreating the curve to enable fast manual modifications. For this reason, a more flexible geometric representation is needed, e.g. a spline with a few well-placed control points, well-placed meaning that their placement should give the best possible accuracy for a given number of control points. Therefore, a tool is developed that takes a wrl file as input and outputs the coordinates for the control points needed to recreate the same curve using the native format of the CAD software.

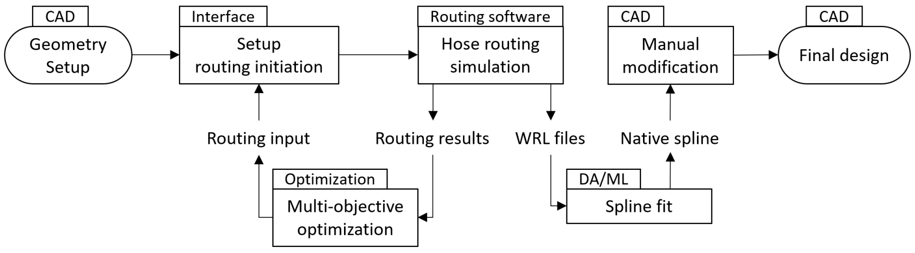

Figure 1. The multi-objective optimization process used as test case.

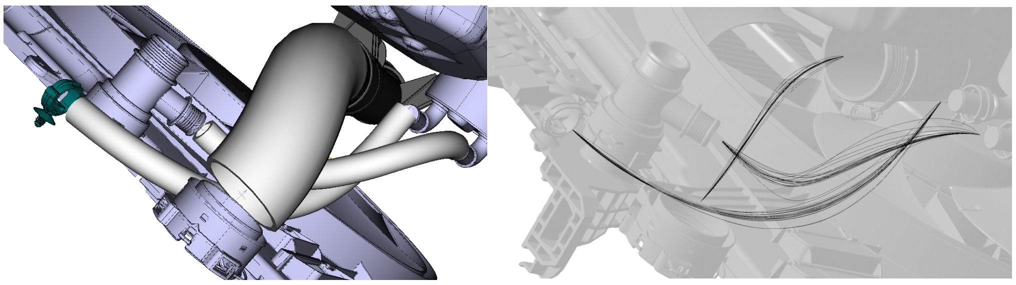

Figure 2. A simulated hose configuration in IPS (left) and the exported center curves for a set of different configurations (right).

For the test case studied, the task is formulated as to create a spline in CATIA v5 using a limited number of control points so that the created spline and the original curve are as similar as possible. Two different ways of accomplishing this task is developed and compared for accuracy and speed. The methods used are a rule-based DA approach using Visual Basics for Applications (VBA) together with CATIA v5, and a data driven approach to predict the control

[1] https://industrialpathsolutions.se/

points based on neural networks (NN) that have been trained using splines created in CATIA v5.

The paper is structured as follows, first some background theory is presented in chapter 2 followed by a presentation of the two approaches in chapter 3. In chapter 4 the results from the two approaches are compared. The paper is then finalized with discussion in chapter 5 and conclusions in chapter 6.

2 Background

The following chapter presents an introduction to design automation and machine learning.

2.1 Design automation

Design automation is used to eliminate non-creative work in the design process through implementation of knowledge based engineering as a means to effectively capture knowledge by storing rules, relations and facts (Amadori et al., 2012). Chapman and Pinfold (Chapman & Pinfold, 2001) describes knowledge based engineering as "an engineering method that represent a merging of object oriented programming (OOP), artificial intelligence (AI) techniques and computer-aided design technologies."

2.2 Machine learning

Machine learning can be summarized as letting a computer make predictions based on past experiences. The overall idea is to model a relationship between a number of inputs and outputs with a mathematical model. In this work supervised learning is used which means the model learns from examples of corresponding input and output values. The supervised algorithms can predict continuous values, called regression, or classify outputs, called class learning. (Baştanlar & Özuysal, 2014)

3 Candidate approaches

To approximate the center curve from IPS in CATIA v5 using its native spline function two approaches are evaluated. A DA approach that iteratively adds control points at the position of maximum deviation between the original center curve and spline in CATIA v5, and an ML approach where a trained model predicts the position of the needed control points, based on training data represented as wrl files.

3.1 The design automation approach

The DA approach uses CATIA v5 with its available VBA API to iteratively fit a spline based on distance measures and the addition of a control point at the position of maximum deviation along the curve, similar to the Ramer–Douglas–Peucker algorithm (Douglas & Peucker, 1973; Ramer, 1972) but using splines instead of line segments. Although many more recent spline fit methods exist for different types of splines and applications (Lin et al., 2018) the Ramer–Douglas–Peucker algorithm is used for this comparison since it is independent of the type of spline used when implemented using DA. It is also intuitive in the way points are added and gives a good visualization of the tradeoff between accuracy and the number control points. The process, seen in figure 3, is handled via the VBA functionality in Excel and direct interaction with CATIA v5. Since the tools in CATIA v5 cannot measure against any geometry in an imported wrl file, the original center curve is first recreated using a spline with control points corresponding to all the points in wrl file, see step 1 and 2 in figure 3. A new spline is then created with control points at the start and end point of the original center curve, starting as a straight line in the first iteration, step 3. The distance, *d,* is measured between the newly created spline and every point in the original curve, step 4. If the termination criteria is not met, another control point is added to the approximative spline at the position of the largest distance, d_{max}, step 5. Step 3 through 5 is then repeated until the termination criteria is met. The termination criteria is controlled by setting a threshold, ε, for maximum deviation, d_{max}, and a maximum number of control points P_{max}, if $d_{max} \leq \varepsilon$ or the number of control points $P \geq P_{max}$ the process is considered complete.

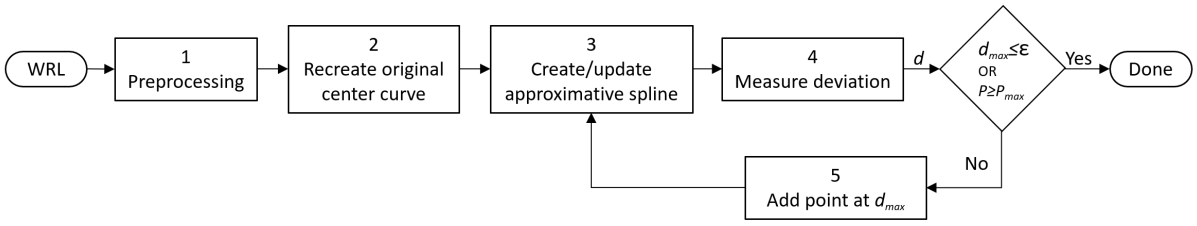
WRL
1
Preprocessing
2
Recreate original center curve
3
Create/update approximative spline
4
Measure deviation
d
$d_{max} \leq \varepsilon$ OR $P \geq P_{max}$
Yes
Done
No
5
Add point at d_{max}

Figure 3. The DA process for spline approximation.

A graphical representation of the process can be seen in figure 4 with the result after step 1 through 3 to the left, followed by three iterations of steps 4, 5 and 3. The recreated original center curve is shown in dashed black and the approximative spline with an orange line. This figure also illustrates the balance between the effort of manipulating the shape of the spline and the accuracy based on the number of control points.

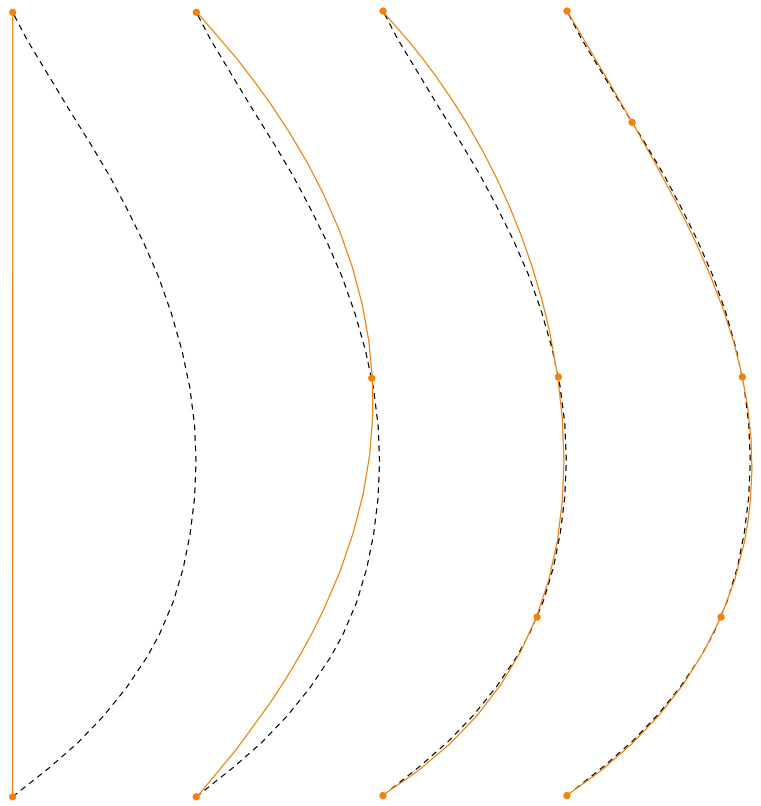

Figure 4. The approximation steps of the DA approach with the original center curve in dashed black and the approximative spline in orange.

3.2 The machine learning approach

The overall idea of this approach is to train an ML model that can predict the best positions of a set of control points for the approximative spline. To accomplish this several different models based on neural networks (NN) (O'Shea & Nash, 2015) and convolutional neural networks (CNN) (O'Shea & Nash, 2015) are evaluated using different architectures and representations of the input and output data. All models are trained using Keras (Chollet & others, 2015) with the Tensorflow 2.0 (Abadi et al., 2015) backend.

Although the data is represented in slightly different ways for the different models used, the general process, visualized in figure 5, stays the same. The model is first trained using annotated data containing both the model input and desired output for a set of examples, steps 1-3 in figure 5. The wrl files used for training are preprocessed into a general format $\overline{d_t}$ containing both the curve and the control points needed to approximate it. The data can then be adapted for a specific model creating the data $\overline{d'_t}$. For the wrl files used when predicting the process is similar, however the target control points are not available in the preprocessed data $\overline{d_p}$, the preprocessing also outputs extra information $\overline{d_{pp}}$ needed in post processing. Just as the training data, the data used for predictions is also adapted depending on the model used creating $\overline{d'_p}$. The predictions made by the model then go through post processing before being written to Excel and used with the VBA API for creation of the approximative spline, step 7-9.

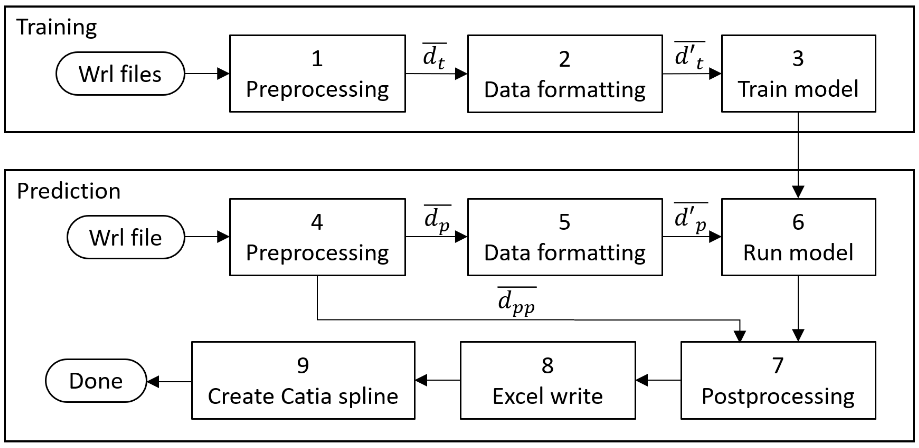

Figure 5. The ML process for spline approximation.

To train the ML models a dataset of examples with both the input to the model and the target output for each example is needed. In this case the dataset needs to contain both the center curve and the control points used to approximate the same center curve with a spline. Since the wrl files from IPS do not contain any information about the needed control points, a training dataset is instead generated using CATIA v5. When exporting a spline to a wrl file from CATIA v5 the file contains both a series of points representing the spline itself, in the same way as center curves from IPS, and the control points used to create it. Since the number of control points needed in a spline to approximate a center curve from the IPS results varies, two sets of splines are generated. One set with four control points and one set with five, each containing 250 000 samples. The splines are generated by creating the target number of points in CATIA v5 with x, y and z coordinates for each point controlled by a separate parameter. A spline is then added using the parametrically controlled points as its control points, enabling the spline to be modified by controlling the parameter values. The VBA API is then used to update the parameters for each point with a random value and export the spline together with its control points to a wrl file. This process is repeated until the target number of wrl files are reached.

Since all the models require a fixed size for the target data, the different number of target control points for the samples creates a problem if the models are used to predict the coordinates of the control points directly. To solve this, two different methods are evaluated:

- Create separate models for each of the two datasets and use a classification algorithm to predict which model should be run.
- N-hot encode the target outputs and treat the problem as a multi-label classification task with each label representing a relative position along the spline, letting the model choose zero or more of these positions to be included in the approximation. N-hot encoding means that instead of using a real value as target the data is represented as a vector of possible solutions, in which n positions can be active i.e. hot.

The generated wrl files are read to extract the coordinates for all points representing the spline (input) and the corresponding control points used to create it (target). Since the positions of all control points for a given spline should be the same relative to the spline itself regardless of scale or position of the spline, the dataset is normalized by scaling and translating each spline individually to fit inside a unit cube. This makes the model insensitive to both position and scale reducing the complexity of the relation between the input and target output. Since the number of points used to represent the spline in each wrl are not consistent and all models also require a fixed input size interpolation is used to calculate coordinates for a set of equally spaced points to be used as input for the models. Several attempts are made to train a model with different number of points used for the input. More points lead to more trainable parameters which increases the difficulty of training the algorithm, fewer points lead to loss of fidelity in the input. From the tests it is concluded that an input with 10 points is the sweet spot for this application. To further decrease complexity the first and last control point for each example is removed from the target since they always correspond to the start and end point of the original spline and therefore do not need to be positioned by the model. To create the n-hot encoded target data, another set of evenly spread points on the spline is needed as a lookup table, representing each possible label for the classification. For this the interpolation tool is used again, with an output of 100 points. The same process is also used on a set of wrl files from IPS to create the IPS data set, in this case excluding the targets and including the scaling factor and translation $\overline{d_{pp}}$. This information is used to de-normalize the model output so that the predicted control points match the original data.

The dataset can be used without much modification to train both an NN and a CNN to predict coordinates for a fixed number of control points (output size) only requiring reshaping of the input vector for the CNN. To classify the examples as needing either 2 or 3 added control points a binary classification approach is used with the classes encoded as 0 or 1 in a single target output value. For the n-hot encoded targets each target output is represented as a vector with 100 elements, each element representing a relative position along the spline at which a control point might be placed. The target control points are marked by a value 1 in the otherwise zero valued vector. To find the relative position of the target control points their coordinates are compared with those of each available relative position found in the corresponding 100-point representation of the spline. Since the index of each 1 in the n-hot encoded targets is easily found, another representation to be used with separate models for the two target sizes is also created using the index of each control point as the target.

To evaluate the performance of the different models on previously unseen data 10% of the samples from the training data is set aside and used as validation set. This is also complemented by the IPS data set consisting of the real IPS data.

First the performance, measured with mean squared error between target and prediction, of an NN and a CNN is compared using the data containing four control points represented as the coordinates for each point. In this comparison the CNN proves to be the better architecture and it is therefore used to build the other models. To compare the different target representations several models are built and evaluated on the validation set using the average deviation between the original and approximative spline. This measure is calculated in CATIA v5 by measuring the deviation at 20 positions along the spline and taking the average value. In table 1 the results for each test are presented as the average over all samples in the validation set. Looking at the performance for the different target representations it becomes obvious that the coordinate representation is underperforming compared to the other representations. The n-hot encoded and point index representations both perform very similar. For the point index representation however, the classifier is also needed to decide which of the models should be used for a specific example. The trained classifier reaches an accuracy of 98% on the validation set which is slightly higher than the n-hot encoded model that reaches 92% regarding the correct number of control points predicted.

Table 1. Average deviation for the different target representations.

Target representation	Average deviation
Coordinates (x,y,z)	10,25 mm
N-hot encoded	4,46 mm
Point index	4,40 mm

When evaluated on the IPS data the classifier used to classify the number of control points needed only predicts four control points although some of the curves would need more control points. This rule out the method of using multiple models and establish the model to be used as a CNN with n-hot encoded targets.

The final model consists of a CNN with two convolutional layers followed by four fully connected layers. The convolutional layers use a kernel size of 3, the smallest kernel able to identify curvature. Each fully connected layer except the output layer uses the ReLu activation function while the output layer uses a sigmoid. Since the target is n-hot encoded the output represents a likelihood of each point being part of the approximation.

The SciPy library find_peaks[2] is used to select the best points to use in the approximation based on the output from the model. To remove some noise from the output data it is sent through a moving average before being fed to the find_peaks function, this together with the options available in find_peaks enables some tuning to get the desired results without retraining the model itself. The model output together with the output from find_peaks is shown in figure 6. With the peaks identified the coordinates for the corresponding points can be found by looking in the 100-point representation of the original spline for the predicted index.

[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html

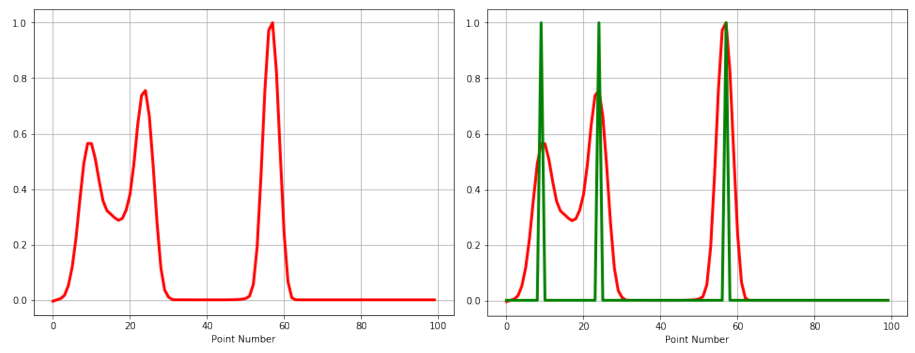

Figure 6. Output from the model (red) to the left and the identified peaks (green) to the right.

When predicting on IPS data the output is noisier and with lower certainty (lower output values on the peaks). It does however still produce peaks and with some scaling and smoothing the output resembles the results from the validation data quite well and can be used with the find_peaks function. Figure 7 shows two examples of the smoothed and scaled model output from IPS data together with the peaks from find_peaks. The corresponding center curves and approximative splines are shown in figure 8.

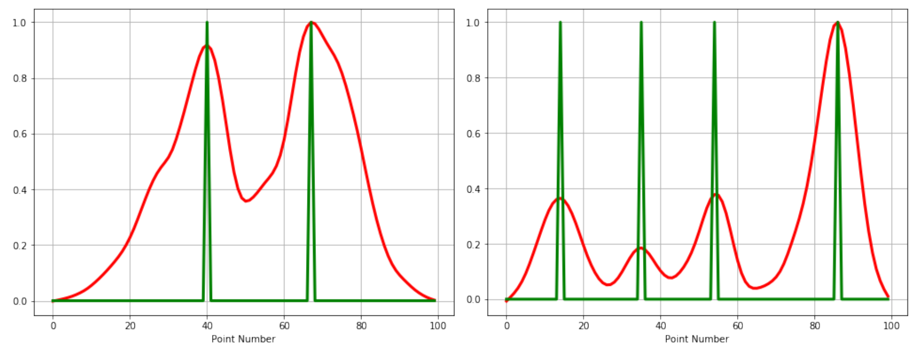

Figure 7. Model output (red) and identified peaks (green) for two test examples.

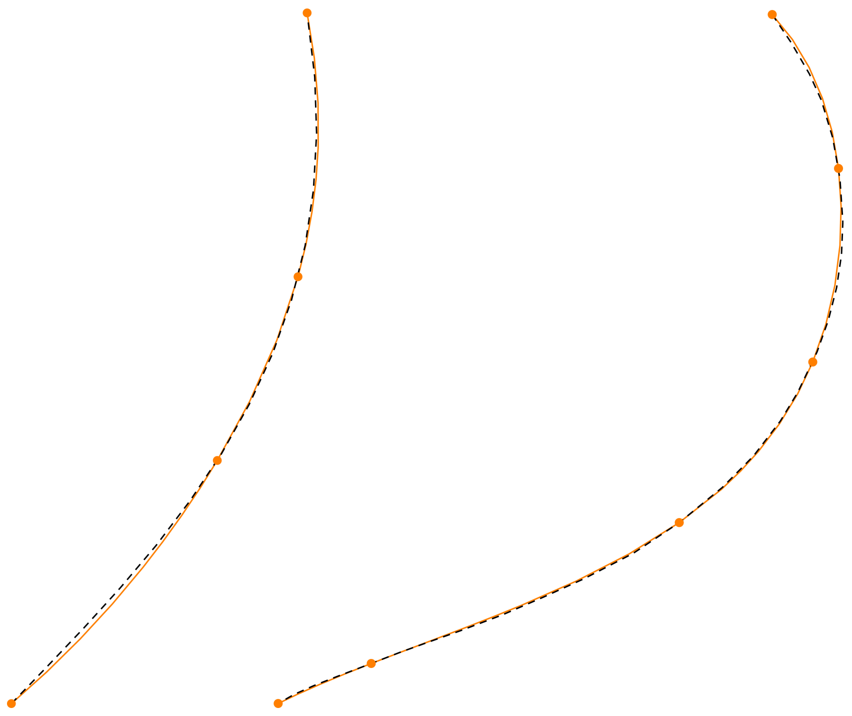

Figure 8. Two examples of center curves in black dashed line with their approximative splines in orange.

4 Comparison of Approaches

The final ML model is compared to the DA approach in terms of accuracy and runtime. A set of 63 wrl files from IPS is used to compare the approaches. The curve length varies between 150 and 350 mm, meaning a deviation of 1 mm represents roughly 0.5 percent of the length.

4.1 Accuracy

The accuracy is evaluated by measuring the distance between the original center curve and the approximative spline for numerous points and taking the average value.

Since the goal is not just accuracy but also to create a curve with few points to make it easily modifiable and the two methods operate very different from each other there is not a single best way to compare their accuracy. Mainly because the ML approach just gives its best guess while the DA approach can be tuned by adjusting the number of points it can add as well as setting a target accuracy in the form of a maximum allowed deviation. To get a good comparison three tests are performed with different settings for the DA approach. These include:

DA 1. Setting the maximum allowed deviation to the same as the one produced by the ML for each example, meaning that the result will be at least as accurate as for the ML

DA 2. Setting the maximum number of points to the same as the result from the ML for all examples

DA 3. A test which produces close to the same average deviation overall.

Results from all the test are represented in table 2 as averages over all samples and the maximum deviation for each sample is shown in figure 9.

Table 2. Test results for the different approaches, all values are averages over all examples.

	ML	DA 1	DA 2	DA 3
Average Deviation	0,35 mm	0,21 mm	0,44 mm	0,35 mm
Max Deviation	0,96 mm	0,56 mm	1,12 mm	0,82 mm
Number of points	5,50	6,22	5,51	5,16

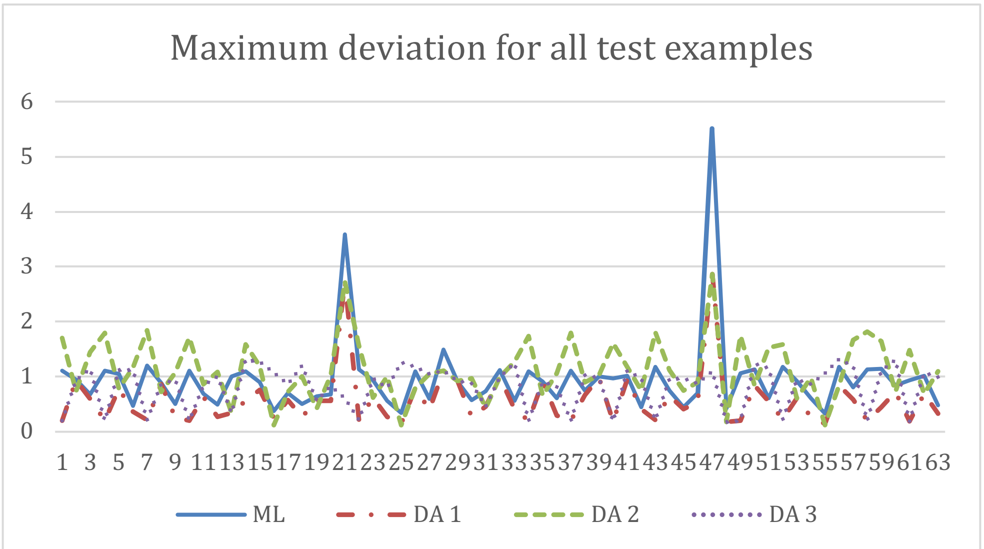

Figure 9. Maximum deviation for all the runs and test samples.

Looking at the maximum deviation for all examples, Figure 9 shows two outliers that are causing problems for the ML algorithm, namely example 21 and 47, which also transfers to DA 1 and 2. Since DA 3 is independent from the ML approach for each example, it is not effected and produces a much more consistent result. Comparing the average values over all examples from table 2 shows that DA 1 produces a better accuracy but at the cost of more points which is to be expected when setting the maximum deviation to the one from the ML, meaning that the DA will produce results that are as good or better with respect to accuracy. When allowing the same number of points, DA 2 produces a lower accuracy than ML indicating that ML makes more efficient use of the available points, this is however contradicted by the fact that DA 3 produces better accuracy with fewer points. The varying results regarding efficient point use indicates that the sample size is simply too small to not be affected by noise in the data. Also, since DA 2 produces quite bad results for examples 21 and 47 it is apparent that the ML algorithm did not get the number of points right and that there should be more points used to get a result in line with the other examples. This is further strengthened by the fact that DA 3 uses 7 points to approximate both the outliers while the ML only uses 4 and 5. This problem might be mitigated by tuning the peak finding algorithm to include more peaks, however this comes at the cost of more peaks on other examples as well, since the settings are global. Overall the performance from both algorithms is very similar regarding efficiency in point use and they can both produce results with an accuracy that is good enough considering that the optimization results are not perfect to begin with which is the reason for approximation.

4.2 Runtime

When comparing the average runtimes to approximate a center curve from the IPS data the ML is the faster alternative by a large margin. The time for DA is 6.8 seconds while the ML takes 0.4 seconds. It should also be noted that the total runtime for the DA approach is dependent on the number of points used in the approximation meaning that there is a lot more variation over the samples.

When comparing runtimes, it should also be noted that the ML approach requires the set up and training of a model beforehand. The single most time-consuming step is to generate and process all the data used for training which in this case consumes around 72 hours. This can however be accomplished without human intervention which also holds true for the training of the model, although that takes just 20 minutes.

5 Discussion

When used to approximate a curve represented as a series of points from a wrl file using an easily modifiable spline both the ML and DA methods reach a satisfactory result. The biggest difference between the two is the relation between speed and flexibility with the DA method being considerably slower but providing a lot more flexibility in terms of tuning for the desired accuracy. The accuracy reached by the ML with an average maximum deviation from the original curve of less than 1 mm is still good enough considering that the shape is to be fine-tuned in the CAD tool. The same can be said about the time consumption for the more flexible DA approach, even though it is 17 times slower than the ML approach the total time to approximate a center curve is still just 7 seconds which is no problem in most cases. Both methods also reach a similar result in the efficiency of point usage, providing roughly the same accuracy for the same number of points. If time is not critical the added flexibility in the DA tool is probably preferable since it can be used for different tasks with different requirements when it comes to accuracy. Also considering the substantial amount of work needed to set up the ML model the time consumption to approximate a curve must be critical for it to be worth the effort.

Although the ML approach did not provide much benefit over the DA, its development lead to some interesting finds regarding the use of ML with engineering data. The CNN proved to be a good algorithm with the input data represented as a few points with x, y and z coordinates. For the test case used 10 points seems to be the sweet spot. The use of convolutions gives the ability to identify geometrical features such as curvature in the data that can then be used to make a prediction. The way data is represented does also generalize to surfaces since the data would have the same structure as an image meaning that the many different CNN architectures available for image data can be applied.

The use of n-hot encoded target data also proved useful giving the ability to handle different target sizes between samples. The n-hot encoded data together with the point index representation also gave considerably better results than using the coordinates for each control point. The reason for this is probably that with both point index and n-hot encoded data the algorithm is forced to select a point on the curve instead of a point in open space, decreasing the amount of possible solutions.

The single biggest performance gain comes from preprocessing the data in an application specific way, in this case normalizing, scaling and translating each example spline individually. Without these steps, using normalization over the complete data set instead, the tests with IPS data gave nowhere near the correct approximation. This is thought to be because the IPS center curves do not share the same distribution of coordinates as the training data, their position is more varied. If enough center curves from IPS were available with the correct approximation these would properly make for better training data. If the generated training data were more closely matching the real data instead of being random the results would probably also be better. However, with random splines as training data the algorithm should in theory be able to approximate every possible spline with the correct number of control points. Data availability is generally a problem, when trying to use data driven approaches like ML in engineering, since large annotated datasets are rarely available. As shown in this case the use of artificial training data does work to a certain extent but creates problems with generalization to the actual data to be used, most notable in that the classifier did not generalize at al. The problem with the classifier might be mitigated by using 1-hot encoded target data so that the classes are not defined by a threshold for a single value but instead as the highest of two separate values. An interesting generalization behavior is noted using the n-hot encoded targets however, namely that the model can predict four added control points even though the training data only contains examples with a maximum of three. This indicates that the model has learned to use the geometrical features to localize areas where control points are needed.

6 Conclusions

Two methods are compared on the task of approximating a center curve represented as a series of many points using an easily modifiable spline with a few well-placed control points. The use case is to enable results from an MOO process to be imported to a CAD tool for manual fine-tuning. With a few, well-placed control points the spline representation enables easy modification by moving the points. The methods compared are a DA approach based on the Ramer–Douglas–Peucker algorithm implemented in a CAD tool using its API and an ML approach using a CNN to predict the best placement of the control points based on the series of points in the original representation. Both methods give a good enough accuracy and speed but show different properties. The DA approach gives a lot of added flexibility since it is an iterative method that can be run until the desired result is reached, this comes at the cost of speed though. The ML method is a lot faster but lacks the flexibility of the DA approach. However, considering that the time to run the DA approach is only 7 seconds, the faster speed of the ML approach would only be beneficial if speed is critical.

Trough the implementation of the ML approach some other findings are made regarding data representation and model architecture. It is shown that a CNN can be used with geometrical input data represented as a number of points containing x, y, and z coordinates and that the algorithm is able to identify geometrical features in the data. As for the target data it is shown that the use of a representation that forces the model to choose one or more points from a predefined set, such as n-hot encoding, instead of using the raw coordinate data gives a lot better performance. The use of a well-researched algorithm such as the CNN that is available for many different uses enables the evaluation of ML techniques on a variety of tasks incorporating geometrical data if a data set is available or can be generated. It is shown that data can be generated if none is available, however this can create problems regarding generalization depending on how well the generated data correspond to the data used to make predictions.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/

Amadori, K., Tarkian, M., Ölvander, J., & Krus, P. (2012). Flexible and robust CAD models for design automation. Advanced Engineering Informatics, 26(2), 180–195.

Baştanlar, Y., & Özuysal, M. (2014). Introduction to Machine Learning. In M. Yousef & J. Allmer (Eds.), miRNomics: MicroRNA Biology and Computational Analysis, 105–128. Humana Press.

Chapman, C. B., & Pinfold, M. (2001). The application of a knowledge based engineering approach to the rapid design and analysis of an automotive structure. Advances in Engineering Software, 32(12), 903–912.

Chollet, F., & others. (2015). Keras. https://keras.io

Douglas, D. H., & Peucker, T. K. (1973). Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. Cartographica: The International Journal for Geographic Information and Geovisualization, 10(2), 112–122.

Lin, H., Maekawa, T., & Deng, C. (2018). Survey on geometric iterative methods and their applications. CAD Computer Aided Design, 95, 40–51.

O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. ArXiv E-Prints.

Ramer, U. (1972). An iterative procedure for the polygonal approximation of plane curves. Computer Graphics and Image Processing, 1(3), 244–256.

Wehlin, C., Persson, J. A., & Ölvander, J. (2020). Multi-Objective Optimization of Hose Assembly Routing for Vehicles. Design 2020 - 16th International Design Conference, Cavtat, October 26-29, 2020.