Optimal Evolutionary Optimization Hyper-parameters to Mimic Human User Behavior

Sneha Saha*, Thiago Rios*, Leandro L. Minku†, Xin Yao†§, Zhao Xu‡, Bernhard Sendhoff* and Stefan Menzel*
*Honda Research Institute Europe GmbH, Carl-Legien-Str. 30, 63073 Offenbach, Germany
†CERCIA, School of Computer Science, University of Birmingham, UK
‡Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China
§NEC Laboratories Europe, Kurfürsten-Anlage 36, 69115 Heidelberg, Germany
Email: {sneha.saha,thiago.rios,stefan.menzel,bs}@honda-ri.de, {l.l.minku,x.yao}@cs.bham.ac.uk, zhao.xu@neclab.eu

***Abstract*—Shape morphing methods are a key representation in human user-centered design as well as computational optimization of engineering applications in the automotive domain. 3D digital objects are modified using deformation algorithms to alter the shape for optimal product performance or design aesthetics. We imagine a system which can learn from historic user deformation sequences and support the user in present design tasks by predicting potential design variations based on currently observed design changes carried out by the user. Towards a practical realization, a large amount of human user deformation sequence data is required which is practically not available. To overcome this limitation, we propose to use a computational target shape matching optimization whose hyper-parameters are tuned to exemplary human user sequence data and that allows us to afterwards generate large data-sets of human-like shape modification data in an automated fashion. In addition, we classified the user sequences to experience levels based on their variance. These user experience-tuned evolutionary optimizers allow us in future to mimic different user behavior and generate a large number of potential design variations in an automated fashion.**

***Index Terms*—evolutionary optimization, similarity measure, representations, clustering, interactive designs**

I. Introduction

Digital development in the automotive domain requires the generation and modification of 3D objects on different levels with different perspectives. Designers e.g. create novel product variations according to given user requirements while engineers focus on providing realizable concepts with optimal technical performance. As a support, computational optimization algorithms, like e.g. evolutionary optimization, are utilized to search for optimal shape parameters minimizing or maximizing given cost functions under constraints. We aim to capture human design processes, i.e. here a sequence of 3D shape modifications carried out with state-of-the-art shape morphing methods, and applying machine learning to build smart time-series models which allow us to predict and generate a variety of 3D design alternatives, which adapt online while the user continuously modifies the 3D shape, along the idea of 2D SketchRNN [1].

However, for training an accurate time-series model, like e.g. a recurrent neural network (RNN) or a long short term memory (LSTM) network, a large sample set of user data needs to be available which is time expensive to generate. To overcome this limit, we propose in the present paper to utilize a computational target shape matching optimization whose hyper-parameters are tuned to exemplary human user sequence data and that would allow us to afterwards generate large sets of human-like shape modification data in an automated fashion. In a first step, we realized a shape deformation tool based on standard free-form deformation (FFD) [2] to record human user interactions for optimizing a given 3D shape towards a target 3D shape. Free-form deformation has been chosen because it is used for manual shape morphing as well as in computational evolutionary design optimization [3]. A conceptual 2D design interaction framework using FFD was proposed by Menzel et al. [4] for predicting design performance based on human user interaction using neural networks. In the second step, we implemented a 3D shape deformation tool to improve on realism and interaction variety wherein human users were asked to perform a simple target shape matching task for a shape primitive. In parallel, a series of computational target shape matching optimizations were carried out based on different hyper-parameter settings for the same task. Both recorded data sets allowed us first to automatically align human user and computational optimization time-series using dynamic time warping for identifying user-specific hyper-parameter settings and, second, to classify different expertise levels among the users based on the variance of their intermediate design steps. Such classification is possible because inexperienced users usually present a higher level of uncertainty and variance in the intermediate steps than experienced users.

The present paper is structured as follows: In Section II, we review literature related to shape deformation techniques, different cluster mechanisms and optimization tasks related to target shape matching. In Section III, we present a 3D shape morphing tool which we implemented to record human user interaction for a simple target shape-matching task. In parallel, we carried out evolutionary optimization based target shape matching which allowed us to align the optimization hyper-parameters to human user data by dynamic time warping. Section IV provides insights into the classification of human users into groups with different experience levels. Finally, Section V concludes the paper.

II. Related Literature

An important aspect for efficient 3D shape manipulation is the choice of representation to minimize the number of design parameters with maximum design flexibility. In CAE, B-splines and NURBS (Non-uniform rational B-splines) play a major role, as they allow the modification of shapes at different scales with few parameters. Shape deformation techniques with free form deformation (FFD) as a most prominent spline-based method directly alter the node points of an underlying geometric mesh which make them very flexible in terms of choosing a reasonable number of parameters and introducing locality on the deformations [6], [7].

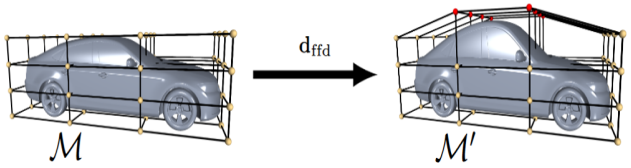


Fig. 1: Free form deformation of a vehicle, taken from [7].

Here, the geometry is embedded in a lattice of control points, the so-called control volume, and the shape is modified and optimized by moving selected control points in 3D space [7]. Zhang et al. [6] show that the defined dimensionality of a problem for shape optimization can restrict the optimal design. Using too few variables may prove certain potential improvements impossible. Conversely, if too many design variables are used, particularly if variables are strongly coupled, the search landscape can become intractably complex to navigate. Regardless of the user-defined parametrization, the final design is most definitely sub-optimal - often limited by parametrization [8]. The design parameter selection for deformation approaches can be seen as a feature selection technique, where the designer abstracts the representation into variables that can be used by both, human users and computational optimizers. But in most cases, the nature of design problems is ill-structured and therefore the designer must engage in defining an abstraction in order to explore design alternatives and optimal solutions. In most cases in the design generation process, designers are actively involved in the co-creation process even without much design knowledge. In our experiments described in Section III, we utilized a minimum number of design parameter variation both for user and optimization runs to reduce the dimensionality and ease the task for the human user. For global industrial design optimization, an evolutionary strategy with covariance matrix adaptation (CMA-ES) [5] is widely used since it promises a good convergence behavior for a low number of function evaluations which is important especially when it comes to simulation-based optimization, e.g. aerodynamic optimization. In CMA-ES, individuals are drawn from a multivariate normal distribution. This set of solutions is called a population based on the history of successful parameter mutations. Each individual in the population is then evaluated using the objective function and the mean and covariance matrix of the normal distribution are updated based on the history of successful parameter mutations. This process is repeated for several iterations, called generations, until a threshold is reached.

Both time-series, human user modifications and optimization parameter variations over the course of generations, have been recorded along with their performance in the target shape matching scenario to evaluate their similarities. Among the different similarity or distance measure of sequences, the most commonly used are Euclidean distance and Manhattan distance [9]. But both measures perform poorly for sequences of uneven length or even when there is distortion in the series. Dynamic Time Warping (DTW) [10] overcomes this drawback and can be applied to sequences with distortions. DTW finds the similarity between two sequences and it finds the optimal match between two sequences. The sequences are ”warped” non-linearly in the time dimension to determine a measure of their similarity independent of certain non-linear variations in the time dimension. This sequence alignment method is often used in time series classification. Besides determining the optimal hyper-parameters of the evolutionary optimization, we aimed to identify user groups of different experience levels based on the time-series history. Typically pattern clustering activities include pattern representation, clustering or grouping, and pattern proximity measure [11], [12]. The output clustering can be hard to partition sequences into a group or it can be fuzzy when each pattern has a variable degree of membership in each of the output clusters. Hierarchical clustering algorithms [13], [14] produce a nested series of partitions based on a criterion for merging or splitting clusters based on similarity measures. Pattern proximity uses a distance function defined on a pair of patterns. The main aim of using clustering is to group the users into subgroups based on their strategy of design deformations. Even if we want to understand exploratory/oscillating pattern and exploitative sequences to get the set of expert designer sequences, an unsupervised way initially will be better to get an idea of the number of clusters in the data-set. Unlike hierarchical clustering, other clustering methods such as DBSCAN [15] are helpful when the sequence data-set is large. But for sparse datasets, DBSCAN cannot cluster sequences with large differences in densities [16], [17]. So, for partitioning sparse amount of sequences, other techniques such as k-means clustering sequences are used by searching for centroids of each cluster repeatedly [18]–[20]. In the next section, we introduce our 3D shape morphing design framework for generating human user sequences and the target shape optimization framework.

III. A Framework for Manual and Computational Target Shape Matching Optimization

Our experimental framework falls into two parts. First, we set up a graphical user interface where the user can modify the shape of a 3D object interactively by using one FFD control point and record the sequence of her/his modifications. Second, we implemented a target shape matching based on CMA-ES for the same deformation scenario to align the

optimization time history with the user sequence for finding optimal hyper-parameters of the optimizer.

A. Graphical Framework

For the interactive human user design tool and the computational target shape matching optimization, we use standard free form deformation (FFD) as geometric representation as it is a common 3D shape modification method in engineering applications. The shape can be altered by 1 control point in x and y direction to allow novice users an easy understanding of the interaction capabilities.

1) Free form Deformation Algorithm: The FFD algorithm used in the experiments has been proposed by Sederberg and Parry [2], where the deformations are calculated using tri-variate Bernstein polynomials. The 3D object is altered by the modification of the parallelepiped control volume which surrounds the shape, i.e., moving one control point modifies the control volume and consequently deforms the object inside. Therefore it can be applied to any kind of complex objects. In the Bezier based FFD method, the deformation is defined in terms of a tri-variate Bernstein polynomial. The displacement Δx of any node $X(s,t,u)$ in the control box is calculated as follows:

$$\Delta \mathbf{x} = \mathbf{B}(s,t,u) \cdot \Delta \mathbf{P} \tag{1}$$

where $\Delta \mathbf{P}$ is the displacement of the node point of the control box and $\mathbf{B}$ is the Bernstein polynomial.

After moving the control point, the shape is deformed and shown to the user, so that s/he can visually estimate which further movements have to be carried out to transform the shape as close as possible to a sphere.

For the experimental set-up, 5 geometries have been created by applying deformation from a sphere primitive using FFD. The shapes are generated by applying a random deformation to one control point of the FFD volume as shown in figures 2 and 3. For the rest of the section, Shape 1 (figure 3a) is used as an example to illustrate the experiments.

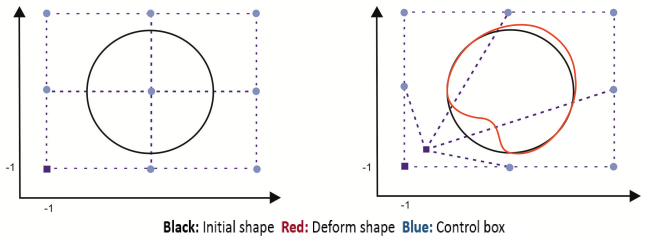
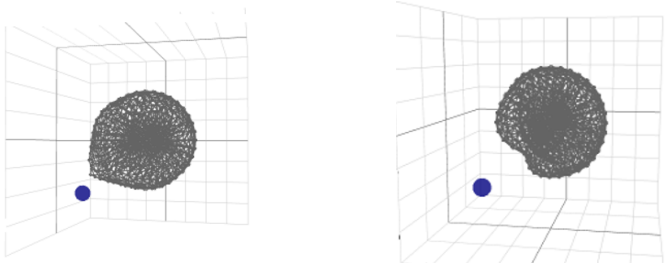


Fig. 2: Shape modification from left initial sphere (black) to Deform sphere (red) by modifying one control point.

A web-based interactive framework is implemented using Dash and Plotly package of python along with FFD techniques. The deformed shaped generated by FFD in figure 2 is used as the initial shape in figure 4 and the control volume is recalculated (figure 4) for the deformed sphere and one of the control point position is initialized at (-1,-1) to perform



(a) Shape 1 (b) Shape 2

Fig. 3: Shapes generated applying FFD to one control point.

the experiments. The human task is visualized in figure 4 where the user needs to modify the initial shape with only one control point (marked in blue) to the target shape shown in the right in maximum 20 steps. The constraints of the control point are such that it can be moved only in 2 dimensions (x and y). If the user reaches the target shape before 20 steps, the last position of the control point sequence is extended to make all the user sequences of equal length and to avoid loss of information due to the variable length of sequences. All iterations of the modification of the design shape are stored and also the geometry at each iteration is stored to evaluate the fitness value. For mathematical formulation, a sequence from design parameter modification in two dimensions (X and Y) from a set of users can be represented as $S = \{S_1, S_2, ...S_i\}$ where $i = 1, 2, ...n$ is the number of users and each sequence is of length 20 denoted by

$$(S(x,y))_i = \{(x,y)_1, (x,y)_2, ..., (x,y)_{20}\} \tag{2}$$

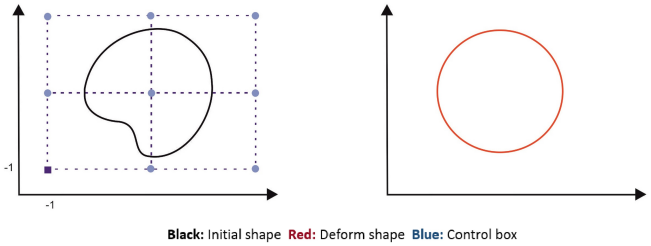


Fig. 4: Schematic overview of the task given to the human to modify initial shape (black) to target shape (red) using only one control point in (x and y direction).

The sequences are collected for 5 shapes by 10 users at present due to time limitation. All the evaluations are done with these user sequences. A detailed analysis of the results is provided for Shape 1. A similar analysis has been done for the other 4 shape variants. When the user interacts with the framework, a sequence according to equation (2) is generated every time. The sequence generated by the variation of the absolute value of the control point position in the design space is plotted in figures 5 and 6.

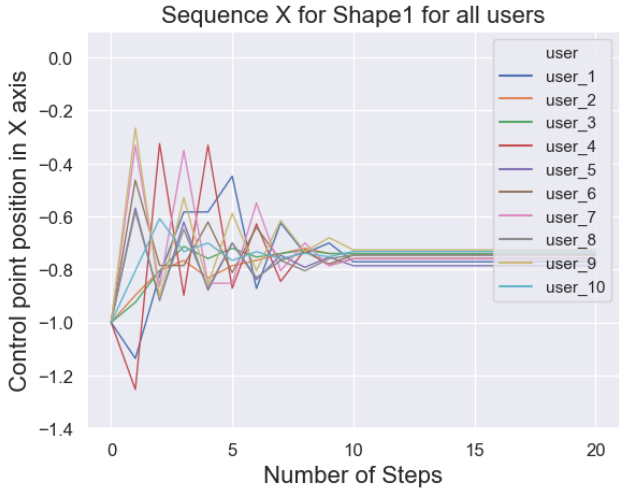


Fig. 5: Plot for sequence variations in X-axis for Shape 1 for all users.

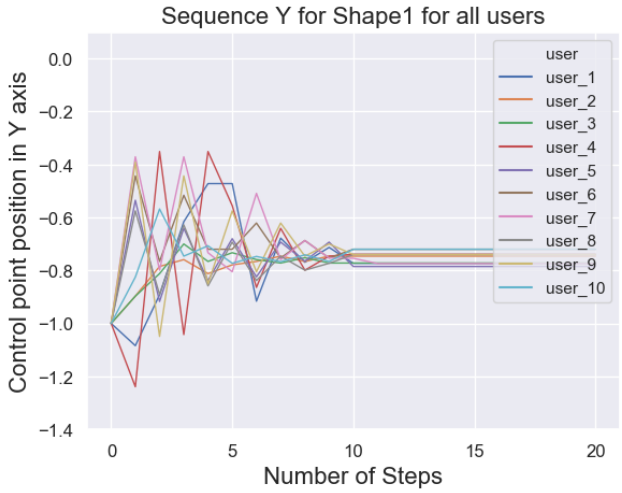


Fig. 6: Plot for sequence variations in Y-axis for Shape 1 for all users.

B. Optimization framework

The selection of the optimizer for the present study accounted for both the character of the problem and the objective of the experiments. The study aims to evaluate the need for the search of optimal hyper-parameter for optimization sequences to align them to each user sequence.

CMA-ES has been used as an evolutionary optimization algorithm due to its suitability for small populations and high convergence ratio. The optimization is realized as a target shape matching scenario which minimizes the distance between an initial given shape (here: the starting sphere (figure 3a)) and a target shape (here: a standard sphere). The approach implemented for the tests follows the proposal in [3], [21]. The parameters for tuning the optimization parameters are discussed in the result section. The objective function should indicate the fitness of the evolving geometry to the target shape. For our case, we considered the fitness function as the modified Hausdorff distance [6] between the initial shape and the target shape to be reached. In CMA-ES, individuals are drawn from a multivariate normal distribution. This set of solutions is called a population. Each individual in the population is then evaluated using the objective function and the mean and covariance matrix of the normal distribution are updated based on the history of successful parameter mutations. This process is repeated for several iterations, called generations. The first step needed is the generation of new individuals, which is done by sampling from a multivariate normal distribution. This is done by the following equation:

$$x_k^{g+1} \sim m^g + \sigma^g \mathbb{N}(0, C^g) \tag{3}$$

for generation $g = 0, 1$ where $x_k^{g+1}$is the individual of $g+1$ generation and m^g is the mean of the search distribution and σ^g is the standard deviation or the step size in the generation.

For each shape, the optimization runs are performed with several combinations of parameter settings which are described in table I.

TABLE I: **Hyper-parameter used for CMA-ES grid search**.

Optimization parameters	Values
Offspring population size	10, 20
Number of parents	2, 3, 5
Total number of generations	20
Initial step size	0.01, 0.1, 1, 1.5, 2

C. Similarity with the optimization runs

To identify the optimal hyper-parameters for the optimization related to a user sequence, we plan to use dynamic time warping (DTW) for its robustness and its ability to handle non-uniform time sequences. The similarity measure is based on the following equations:

$$(S(x_1, y_1))_1 = \{(x_1, y_1)_1, (x_1, y_1)_2, ..., (x_1, y_1)_{20}\} \tag{4}$$

$$(S(x_2, y_2))_1 = \{(x_2, y_2)_1, (x_2, y_2)_2, ..., (x_2, y_2)_{20}\} \tag{5}$$

$$DTW(X, Y) = min(W(x_i, y_j)) \tag{6}$$

The sequence S_1 and S_2 can be arranged to form an $n \times n$ plane or grid where each grid point corresponds to an alignment between elements $(x_1, y_1)_i$ and $(x_2, y_2)_i$. A warping path W maps or aligns the elements of S_1 and S_2 such that the distance between them is minimized. The distance measure between two elements of the sequences is done by Euclidean distance. So, to find the most suitable hyper-parameter setting for each user, we calculated the similarity measure for each sequence of the users with the different combination of (μ, λ) and $step_{size}$ for the optimization runs for shape matching. Each of the optimization runs is the average of 20 generations. So, the final solution sequence is considered as the average of all the generations. The similarity measure is performed to match each of the user sequences with the optimization solutions. The heat map of the DTW similarity measure is shown in

TABLE II: **Optimization parameter similarity with the human user**.

Users	Optimization parameters			DTW
	μ	λ	step	
User 1	2	10	1.5	0.39
User 2	2	10	0.1	0.08
User 3	2	10	0.1	0.12
User 4	2	10	1	0.55
User 5	2	10	0.1	0.44
User 6	3	10	0.1	0.38
User 7	5	20	1	0.50
User 8	2	10	0.1	0.36
User 9	2	10	1	0.46
User 10	3	10	0.1	0.15

figure 7 to give a visual impression. Each row corresponds to a sequence of the optimization run with different optimization parameters given in form of (μ, λ, $step_{size}$). A darker color indicates a higher similarity between the user and the solution of the optimization runs and vice versa. Optimal parameter

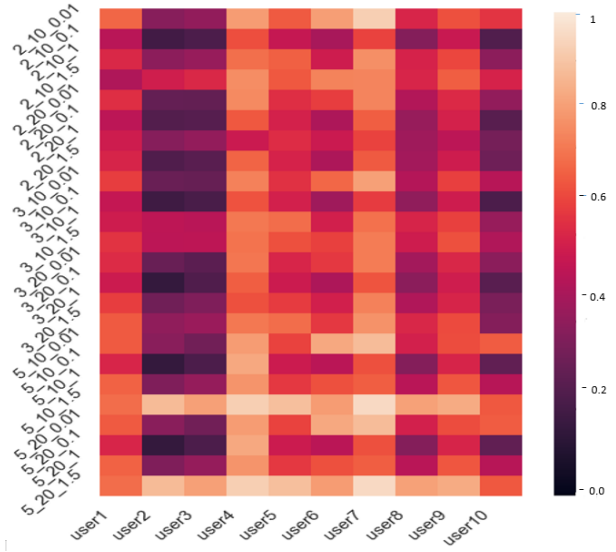


Fig. 7: Heat map of the DTW comparison of the users and optimization runs (The notation on the y-axis is: (μ, λ, $step_{size}$)).

settings for each of the users are given in table II. The parameters are selected by selecting the runs which result in the least DTW measure.

To visually analyze the similarity of the heat map plots, we compared the plot of user $user_2$ modification sequence with the optimization sequence using the hyper-parameters given in table II, ($\mu = 2, \lambda = 10$, $step_{size} = 0.1$) along both X and Y axes in figure 8 and 9. Also, for the comparison we added the plot for hyper-parameters ($\mu = 5, \lambda = 10$, $step_{size} = 0.1$), as the DTW measure with $user_2$ was less for this sequence.

In our current approach, we carried out a grid search on

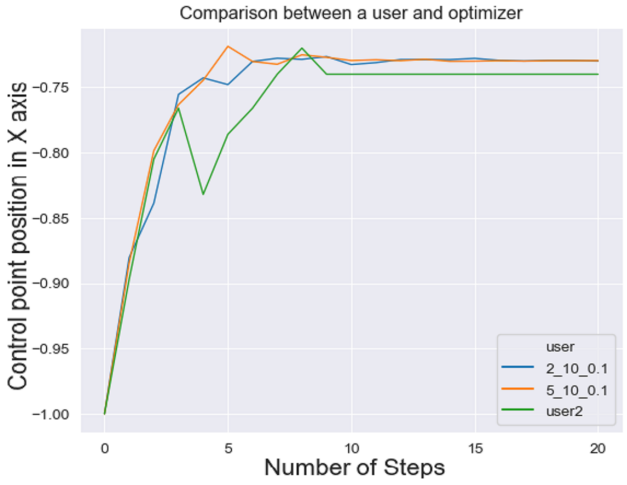


Fig. 8: Comparison of the human sequence with the optimization sequence based on minimum DTW distance - X axis ($user_2$).

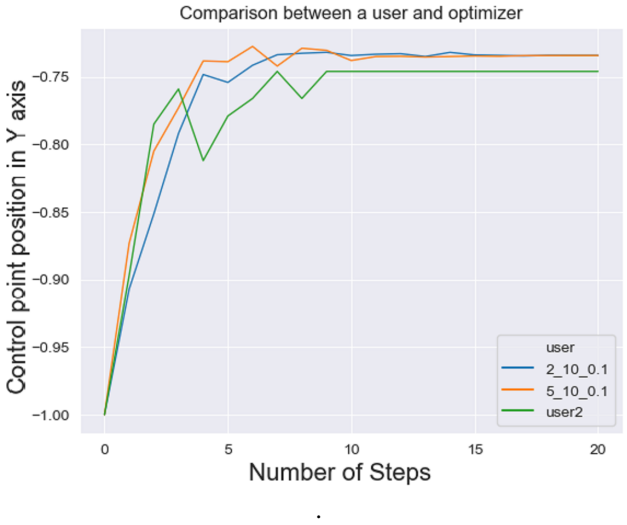


Fig. 9: Comparison of the human sequence with the optimization sequence based on minimum DTW distance - Y axis ($user_2$).

different optimization parameter settings for extracting the optimization sequence and comparing it to the human user sequence by DTW. However, in our next step, we want to extend this process by automatic hyper-parameter tuning algorithms to improve the similarity with a more accurate variation of parameter settings. From figure 7, we see that different users have similar sequence patterns while others are very different. This leads us to group different users together based on their experience levels, instead of comparing each user sequences with a range of optimization sequences of different parameter settings. Furthermore from the heat map in figure 7, we see that the color intensity column-wise for few users are similar to some of the parameter settings. This can be due to the fact that those set of users sequences have some

similarity. For example, in $user_2$, $user_3$, $user_{10}$ sequences, the color intensity for many of the optimization parameter settings are similar. Thus, to estimate the similarity between the users to understand their level of experience for design modification we did further clustering of the user sequences.

IV. User Clustering According to Experience Level

To understand human behavior of shape modification and classify user based on their experience level, we first calculate the similarity between the users. The similarity between the users is also measured by the DTW distance matrix. The distance matrix visualization is done with a heat map shown in figure 10, where each row and column represents a user.

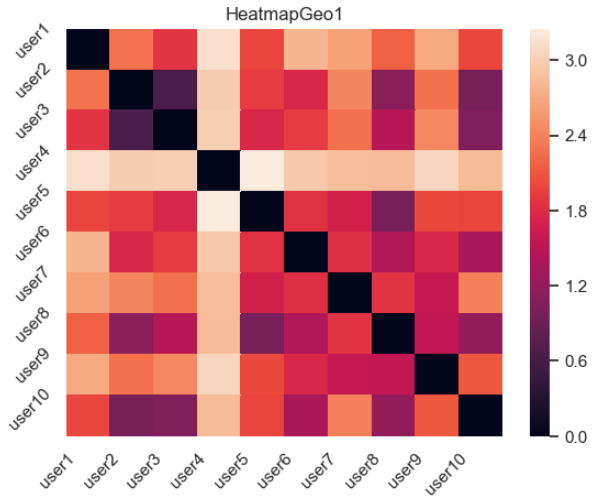


Fig. 10: Heat map of DTW similarity measure.

From the heat map, it can be seen that $user_3$ and $user_2$ has maximum similarity in terms of behavior, while the modification pattern of $user_4$ is very different. Thus to understand the pattern of exploration for each sequence, first, we used the average of the variance in each of the X and Y axes. The variance distributions of each user are shown in figure 11.

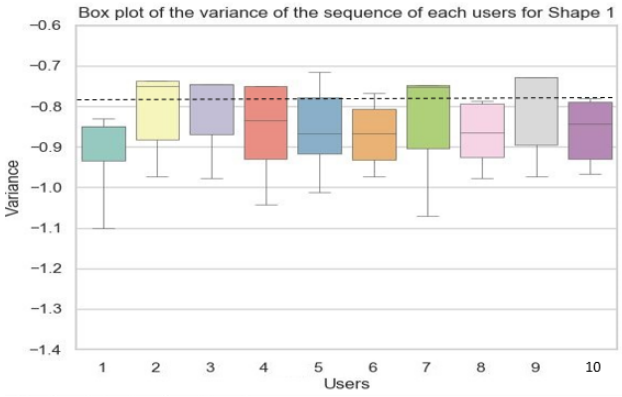


Fig. 11: Plot of the variance of the 10 users for Shape 1.

Since the variance of a sequence measures the distance of each value in the sequence from the mean, the amount of exploration with the design parameters results in different variances for the sequences. Based on the variance of each sequence as an expression of user experience, we used standard k-means clustering to separate the users into groups in an unsupervised way. In order to determine the number of clusters, one method is to calculate the sum of square error (SSE) for a range of values of k. Figure 12 indicates that the optimal number of k in our case is $k = 3$.

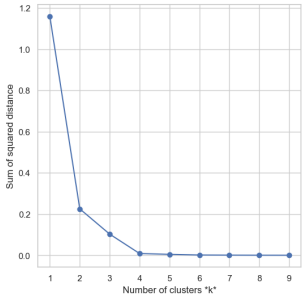


Fig. 12: Number of clusters for k-means.

K-means clustering is performed for all the sequences of Shape 1, and the sequences are colored based on the labels of the k-means clustering algorithm. The plot for the clustering result of all user sequences along the X dimension is shown in figure 13.

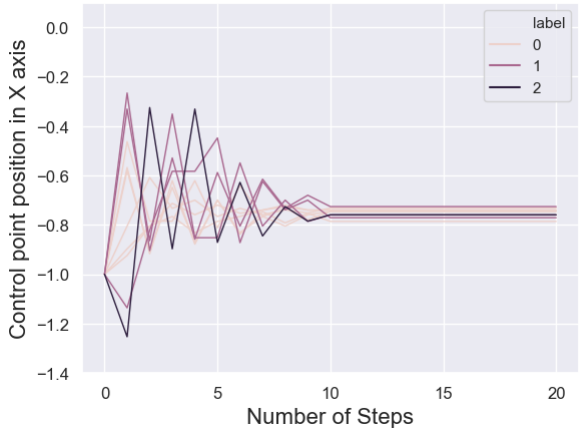


Fig. 13: Clustering based on k-means according to user groups.

However from visual perception, some of the oscillating sequences are labeled unintuitively, e.g. we expected to have $user_7$ and $user_9$ in one group with $user_4$ as inexperienced user class. To improve the clustering, we calculated for each user sequence the sum of the consecutive differences between all the elements along the sequence which is a different way to estimate the modification variations along the sequence. The assumption for this measure is that the experienced users with knowledge of the design space and FFD will modify the design parameter in a monotonic manner rather

than an inexperienced user who would search in the design space in an exploration or oscillating path before figuring out the optimal target position for shape matching. The total variation is represented by C_{index}, which is the measure of the sum of the absolute value of the consecutive differences between the elements of a sequence. Thus, the sum of the consecutive change C_{index} is given below in equation (7)-(9), where the numerator is the sum of the absolute value of the consecutive discrete differences of the elements of a sequence. The denominator normalizes the effect of different overall distances between start and endpoint. The X and Y sequence for each user is given as $(S(x))_i = \{x_1, x_2, ..., x_{20}\}$ and $(S(y))_i = \{y_1, y_2, ..., y_{20}\}$. For each axis, the sum of consecutive differences are calculated separately and we take the average of both.

$$\mathbb{C}_x = \frac{\sum |x_{i+1} - x_i|}{|x_{20} - x_1|} \tag{7}$$

$$\mathbb{C}_y = \frac{\sum |y_{i+1} - y_i|}{|y_{20} - y_1|} \tag{8}$$

$$C_{index} = \frac{(\mathbb{C}_x + \mathbb{C}_y)}{2} \tag{9}$$

Figure 14 shows the calculation of the C_{index} for two different examples of user sequences, where the change at each step for the linear sequence is much less than the step change at each point for the oscillating sequence. Thus, the change $\sum |\Delta x'_i|$ is much higher than $\sum |\Delta x_i|$ where $i = 1, 2...20$.

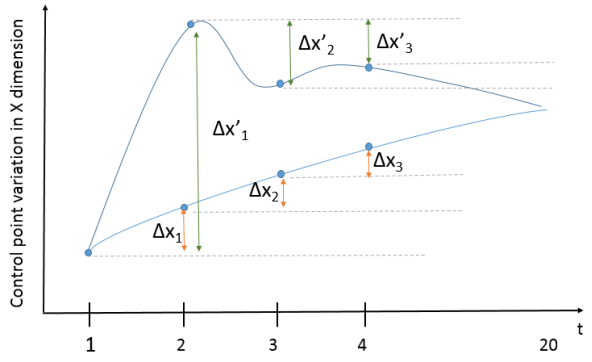


Fig. 14: Sum of the sequence variations along each axis.

A k-means clustering with $k = 3$ is done on the C_{index} value calculated above in equation (9). Figure 15 shows the plot for clustering results of all user sequences along X dimension. From visual perception, the result of clustering based on the sum of consecutive difference two elements of along the sequence looks more promising, so we plan to use this estimation for further classifications. Also, to confirm the above k-means clustering results, another way of unsupervised clustering using a hierarchical algorithm yields a *dendrogram* representing the nested grouping of patterns and similarity levels at which the grouping changes. The height of the dendrogram in figure 16 indicates the order and distance in which the clustering is done. The key point of the above

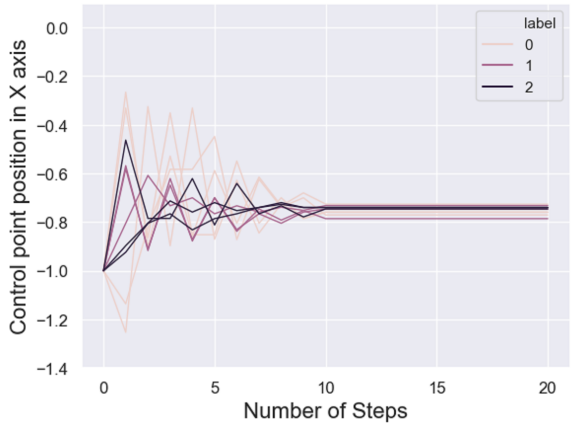


Fig. 15: Clustering based on k-means label (Sum of consecutive difference along a sequence).

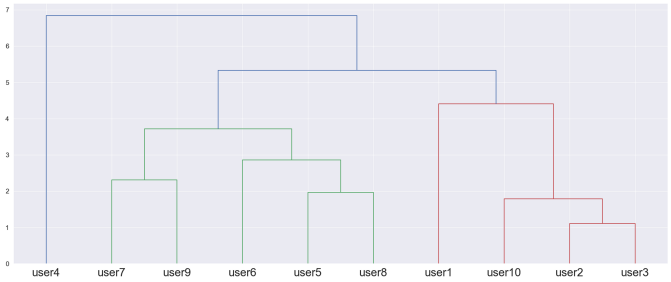


Fig. 16: Hierarchical clustering of user sequences.

process is to replace two objects by a cluster and still be able to define the distance between such clusters. The different color scheme of the dendrogram and the vertical distance from the top of dendrogram indicate the distance or the dissimilarity between clusters. The results of the clustering labels of the user sequences in figure 5 and 6 are explained in table III.

TABLE III: **Clustering labels for Shape 1 modification by the users**

Users	C_index	K-mean	Hierarchical
User 1	6.76	**Label 1**	**Label 0**
User 2	1.21	**Label 0**	**Label 0**
User 3	1.53	**Label 0**	**Label 0**
User 4	12.96	**Label 2**	**Label 2**
User 5	6.88	**Label 1**	**Label 1**
User 6	4.23	**Label 1**	**Label 1**
User 7	9.25	**Label 2**	**Label 1**
User 8	5.06	**Label 1**	**Label 1**
User 9	9.02	**Label 2**	**Label 1**
User 10	2.21	**Label 0**	**Label 0**

From the results of the two clustering labels for Shape 1, out of 10 users, the labels for 7 users are the same for both of the clustering schemes. $user_2$, $user_3$ and $user_{10}$ has similar labels for both of the clustering results and C_{index} is also

low in table III. So we can conclude that these are the most experienced users for our experimental setup. $user_4$, $user_7$ and $user_9$ are the explorative users because they have higher C_{index} values in reference to figure 5.

In the next step, we use C_{index} for the sequences generated by the optimizer for different parameter settings. We calculated C_{index} for the mean of solution sequences of the optimizer over 20 generations. Table IV shows the C_{index} value for different optimization parameter combinations i.e. for different (μ, λ) and initial step sizes.

TABLE IV: C_{index} **values for optimization runs**

	(μ, λ)					
step size	**(2,10)**	**(2,20)**	**(3,10)**	**(3,20)**	**(5,10)**	**(5,20)**
0.001	1.27	1.41	1.22	1.3	1.40	1.04
0.1	1.10	1.89	1.15	1.14	1.23	1.18
1	1.81	4.94	3.94	3.6	3.8	2.3
1.5	5.58	2.21	7.12	4.24	5.52	5.79
2	15	3.07	3.78	2.34	3.9	4.33

In the following, we compare optimizer sequences with the explorative users - $user_4$, $user_7$ and $user_9$ from table III. We identify two optimizer parameter combinations in table IV with highest C_{index} values. The C_{index} values for optimizer parameter combinations ($\mu = 2, \lambda = 10$, $step_{size} = 2$) and ($\mu = 3, \lambda = 10$, $step_{size} = 1.5$) are the highest. So, a graphical plot is depicted for the explorative users and the chosen optimizer parameters in figure 17.

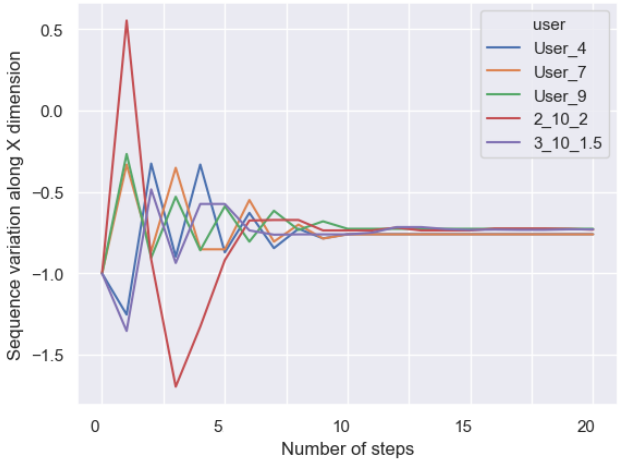


Fig. 17: Solution sequence for x - explorative users and optimizer sequences.

Thus from figure 17, sequence of optimizer parameter setting ($\mu = 2, \lambda = 10$, $step_{size} = 2$) matches with the explorative human user sequences. Another sequence in figure 17 (red line) of optimizer parameter setting ($\mu = 3, \lambda = 10$, $step_{size} = 1.5$) is way more oscillating than the previous optimizer sequence.

Similarly, we also compare the users with low C_{index} value with the optimizer parameter settings with low C_{index} value from table IV. From the table III, $user_2$, $user_3$ have low C_{index} values. So in figure 18, we compare these users with corresponding optimization parameters ($\mu = 2, \lambda = 10$, $step_{size} = 0.1$) and ($\mu = 3, \lambda = 10$, $step_{size} = 0.1$) which have low C_{index} values.

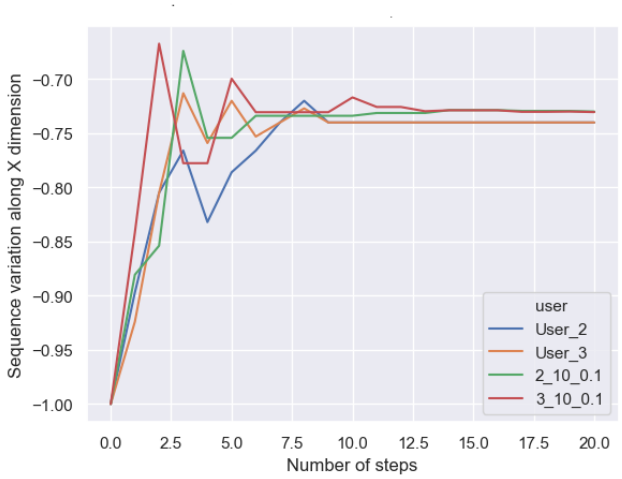


Fig. 18: Solution sequence for x- expert users and optimizer sequences.

From figure 18, sequences for both human and optimization parameters look nearly similar. Thus, we could match both explorative and expert users with appropriate optimizer parameter settings. In the future, we want to compare user behavior with a set of optimizer parameter settings for more complex shapes.

V. Conclusion and Future Work

In the present paper, we address the process of generating large amounts of human user-like design deformation sequences in a monotonic manner that is required for a potential training of recurrent neural networks in the digital development of engineering applications. To mimic the human user behavior, we propose to find the optimal hyper-parameters of an evolutionary optimizer based on a parameter grid search and dynamic time warping technique. We implemented a simplified interactive target shape matching tool based on free form deformation to record human user data sequences. In a similar fashion, we ran several target shape matching optimizations using CMA-ES with varying hyper-parameters. We showed that we can utilize dynamic time warping successfully to identify the optimal hyper-parameters for an evolutionary optimization which mimics the human user sequences. These tuned optimizers allow an application in a wider range of target shape matching optimizations to foster larger sets of deformation sequences. In a future work, we want to generate also more human user data and implement a hyper-parameter optimization framework to overcome the grid search or even automatically construct optimizers of different

types than CMA-ES for best sequence matches. In a second step, we showed that we can successfully cluster human user sequences into groups with different levels of experience. The consequences are twofold. First, large amounts of data according to different user levels can be generated to train different recurrent models reflecting these user states, and second, we can utilize these models to coach inexperienced users by supporting them with models of higher user levels.

ACKNOWLEDGEMENT

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 766186.

REFERENCES

[1] D. Ha, and D. Eck, "A neural representation of sketch drawings," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, Canada*, 2018.

[2] T. W. Sederberg, and R. S. Parry, "Free-form deformation of solid geometric models," *ACM Siggraph Computer Graphics*, 20(4), pp. 151-160, 1986.

[3] S. Menzel, and B. Sendhoff, "Representing the Change - Free Form Deformation for Evolutionary Design Optimization," in *Evolutionary Computation in Practice*, Springer, pp. 63-86, 2008.

[4] S. Menzel, M. Olhofer, and B. Sendhoff, "A meta model-driven interactive framework for design assistance system," in *DS 60: Proceedings of DESIGN 2010, the 11th International Design Conference, Dubrovnik, Croatia*, pp. 1371–1380, 2010.

[5] N. Hansen, and A. Ostermeier, "Completely Derandomized Self-Adaptation in Evolution Strategies," *Evolutionary computation*," 9(2), pp. 159-195, 2001.

[6] P. Zhang, X. Yao, L. Jia, B. Sendhoff, and T. Schnier, "Target shape design optimization by evolving splines," in *IEEE Congress on Evolutionary Computation, CEC 2007, Singapore*, pp. 2009-2016. 2007.

[7] D. Sieger, S. Menzel, and M. Botsch, "On Shape Deformation Techniques for Simulation-Based Design Optimization," in *New Challenges in Grid Generation and on Adaptivity for Scientific Computing*, Springer International Publishing, pp. 281-303, 2015.

[8] R. G. Anderson, and M. Artemis, "Adaptive Shape Parameterization for Aerodynamic Design," in *Structures, Structural Dynamics, and Materials Conference, Kissimmee, USA*, 2015.

[9] J. Yu, J. Amores, N. Sebe, P. Radeva, and Q. Tian, "Distance Learning for Similarity Estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3), pp. 451-462, 2008.

[10] E. Keogh, and A. C. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and information systems*, 7(3), pp. 358-386, 2005.

[11] C. R. Dubes, and A. K. Jain, "Algorithms for clustering data", Prentice-Hall, 1998.

[12] G. Gan, M. Chaoqun, and W. Jianhong, "Data clustering - theory, algorithms, and applications," Society for Industrial and Applied Mathematics, 2007.

[13] S. C. Johnson, "Hierarchical clustering schemes", Psychometrika, 32(3), pp. 241-254, 1967.

[14] Z. Nazari, M. Nazari, and D. Kang, Dongshik, "A Bottom-up Hierarchical Clustering Algorithm with Intersection Points," *International journal of innovative computing, information and control: IJICIC*, 15(1), pp. 291-304, 2019.

[15] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, USA*, pp. 226-231, 1996.

[16] S. H. Khaing, and T. Thein, "An efficient clustering algorithm for moving object trajectories," in *Proceedings of the 3rd International Conference on Computational Techniques and Artificial Intelligence, Singapore*, pp. 11-12, 2014.

[17] Z. Shao, and Y. Li, "On Integral Invariants for Effective 3-D Motion Trajectory Matching and Recognition," *IEEE Transactions on Cybernetics*, 46(2), pp. 511-523, 2015.

[18] N. Ferreira, T. J. Klosowski, E. C. Scheidegger, and T.C.Silva, "Vector Field k-Means: Clustering Trajectories by Fitting Multiple Vector Fields," *Computer Graphics Forum*, 32(1), pp. 201-210, 2013.

[19] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, "A system for learning statistical motion patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9), pp. 1450-1464, 2006.

[20] J. Melo, A. Naftel, A. Bernardino, and J. Santos-Victor, "Detection and classification of highway lanes using vehicle motion trajectories," *IEEE Transactions on Intelligent Transportation Systems*, 7(2), pp. 188-200, 2006.

[21] N. Hansen, Y. Akimoto, and P. Baudis, CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, 2019.