Hindawi
Complexity
Volume 2019, Article ID 3509263, 16 pages
https://doi.org/10.1155/2019/3509263

WILEY | Hindawi

Research Article

Evolving Stencils for Typefaces: Combining Machine Learning, User's Preferences and Novelty

Tiago Martins, João Correia, Ernesto Costa, and Penousal Machado

CISUC, Department of Informatics Engineering, University of Coimbra, 3030 Coimbra, Portugal

Correspondence should be addressed to Tiago Martins; tiagofm@dei.uc.pt

Received 19 August 2018; Revised 28 January 2019; Accepted 4 March 2019; Published 26 March 2019

Guest Editor: Jon McCormack

Typefaces have become an essential resource used by graphic designs to communicate. Some designers opt to create their own typefaces or custom lettering that better suits each design project. This increases the demand for novelty in type design, and consequently the need for good technological means to explore new thinking and approaches in the design of typefaces. In this work, we continue our research on the automatic evolution of glyphs (letterforms or designs of characters). We present an evolutionary framework for the automatic generation of type stencils based on fitness functions designed by the user. The proposed framework comprises two modules: the evolutionary system, and the fitness function design interface. The first module, the evolutionary system, operates a Genetic Algorithm, with a novelty search mechanism, and the fitness assignment scheme. The second module, the fitness function design interface, enables the users to create fitness functions through a responsive graphical interface, by indicating the desired values and weights of a set of behavioural features, based on machine learning approaches, and morphological features. The experimental results reveal the wide variety of type stencils and glyphs that can be evolved with the presented framework and show how the design of fitness functions influences the outcomes, which are able to convey the preferences expressed by the user. The creative possibilities created with the outcomes of the presented framework are explored by using one evolved stencil in a design project. This research demonstrates how Evolutionary Computation and Machine Learning may address challenges in type design and expand the tools for the creation of typefaces.

1. Introduction

Typefaces are an essential resource employed by graphic designers [1], who are always willing to experiment with type and to explore new thinking, tools, and techniques. However, the creation of a typeface is a laborious process, involving the design of several glyphs for different characters. In the domain of type design, a glyph consists in a particular design of a character, e.g., a letter, figure, or punctuation mark. This, along with the increasing demand for new type design work, increases the need for good technological means to assist the designer in the creation of a typeface.

Although conventional computational design tools are effective for precise design tasks during the later phases of the design process, they offer insufficient support to design exploration during the earliest, essentially conceptual, stages of the design process. We also consider that most of the prominent software design tools tend to bias and limit the designers, who become accustomed to work and think in terms of the primitives that these tools provide, the workflow they induce, and the boundaries, implicit or explicit, that they establish. As a result, the outcome of the design project tends to be, at least partially, shaped by the tools, leading to visual tendencies. Therefore, we argue that it is as important to master and exploit the tools at hand, as it is to challenge those tools, by modifying them or inventing new ones that suit unique ideas and design projects.

In this work, we explore an evolutionary approach for the computational generation of glyphs. This approach is intended to provide the designer with a wide range of alternative designs as stimuli for inspiration, working in a mind-opening way and promoting new ideas. We do not expect our approach to competing with more traditional type design approaches, or to replace the designer. Our goal is to develop a tool that aids the designer.

Although some evolutionary approaches for type design exist [2–8] most of them rely on user evaluation, *i.e.* make use of Interactive Evolutionary Computation (IEC). Although

asking the users to evaluate the designs being evolved enables them to directly influence the course of the evolution, this approach puts a considerable burden on them. This leads to user fatigue and, consequently, to the inefficient exploration of the search space. In addition, some of the identified evolutionary approaches require pre-existing typefaces or skeletons, the drawing of initial seed glyphs, or the identification of letter parts.

In the work *Evotype,* we have been combining Evolutionary Computation (EC) and Machine Learning (ML) to evolve glyphs in an autonomously way, with automatic fitness assignment. We started with a Genetic Algorithm (GA) that evolved different populations of candidate glyphs, one per target character, with and without migration of glyphs between populations [9, 10]. Although these early approaches were already able to evolve glyphs with expressiveness and legibility, the glyphs often lacked coherence. In other words, the evolved glyphs had no common visual structure and for this reason, they did not seem to be part of a single typeface. We addressed this limitation by evolving one population of individuals, each being able to express all the glyphs [11]. Each individual consists of a stencil composed of lines that can be used to construct glyphs. This approach provided more coherence to the final glyphs, since their structure share elements of the stencil. The fitness assignment was autonomous, and it was able to guide evolution towards stencils that produce simple, legible and coherent glyphs.

In this paper, we expand our approach to type stencil design by:

(i) Developing an ML approach to evaluate the glyphs produced by evolved stencils, combining a Convolutional Neural Network (CNN) with Self-Organising Maps (SOMs) to evaluate their recognisability as the target character and similarity to existing glyphs, respectively;

(ii) Changing the genetic representation of each stencil to enable the encoding of Bézier curves, and this way provide more expressiveness to the stencil;

(iii) Adopting an approach similar to the one developed by [12, 13], allowing the users to design fitness functions through a responsive user interface, thus allowing them to express their design intentions at the meta-level. The user-designed fitness functions are based on features presented by the stencils, namely behavioural features, related to how each stencil performs in drawing glyphs for the target characters, and morphology features, related to the structure and components of each stencil;

(iv) Based on previous experimental results, as the evolutionary process unfolds, the stencils being evolved tend to converge towards an optimum, resulting in visually similar stencils. The lack of diversity problem is not new in the domain of arts and design, and have been addressed by novelty search algorithms by several authors in robotics [14], art [14, 15] and games [14]. We employ an archive mechanism with hybrid tournament selection [16] that allows us to address this issue, promoting diversity among the stencil being evolved and providing a way to summarise the evolutionary runs.

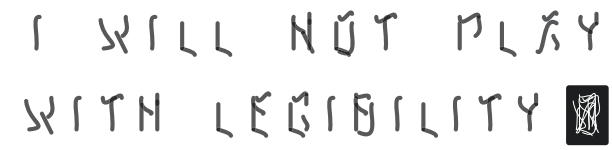

Figure 1: Stencil, and its glyphs, evolved with the presented framework in experiment I.

The experimentation described herein focuses on validating the novel aspects of the approach. We begin by assessing the adequacy of the representation and of the ML-based fitness components by performing experiments on the evolution of stencils that are compact, composed of lines with continuity between them, and that produce glyphs that are recognisable and similar to existing typefaces. Then we test the user interface by performing and analysing runs with user-defined fitness functions. The analysis of the experimental results aims at two core aspects: the ability of the GA to optimise fitness and the ability of the evolved stencils to (i) capture the design preferences expressed by the users and (ii) meet their expectations. Finally, we assess the ability of the novelty search mechanism to generate diverse stencils in a single evolutionary run and the adequacy of the archive, produced during the process, to summarise the results that are then presented to the user.

Overall, the experimental results (see Figure 1) show the adequacy of the proposed approach, demonstrating how EC and ML may address challenges in type design and expand the tools for the optimisation of the design process. Additionally, they also show how meta-level interactive evolution [12, 13] allows the expression of user intentions and goals without imposing a burden to the user, and how novelty search increases the diversity of feasible solutions.

The remainder of this paper is structured as follows. First, we overview the proposed framework. Then, we conduct experiments on the framework, describing the experimental setup and analysing the experimental results. Finally, we present conclusions and directions for future work.

2. Approach

Similarly to our previous work [11], the presented approach is based on the idea of a stencil capable of generating every letter of the alphabet in a coherent manner. In 1876, the American engineer Joseph A. David developed the *Plaque Découpée Universelle* (see Figure 2). This stencil consists of a grid of lines that enables the construction of letters, numbers, punctuation, accents, etc. [17]. The seven-segment display, invented a few decades later, employs a similar approach as the PDU by switching on and off its segments in different combinations in order to represent figures and letters.

The design of typefaces typically involves the creation of modular parts that are then combined by the designer to form different glyphs. By careful looking at the glyphs of

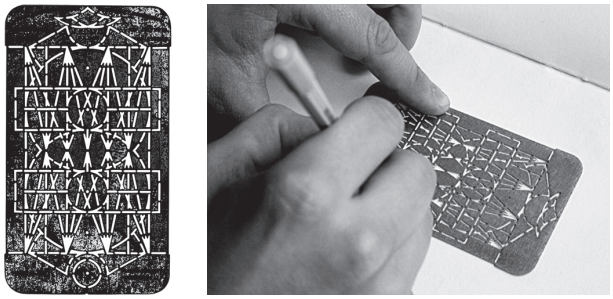

Figure 2: *Plaque Découpée Universelle*, Joseph A. David, 1876.

a given typeface, one may understand their anatomy [18], i.e. the reuse of smaller parts among glyphs. This sharing of parts between glyphs is fundamental to provide them visual coherence. Similarly, the elements (e.g. lines) that construct a stencil can work as a unifying grid that also provides coherence to glyphs created with those elements.

We present a system that employs a Genetic Algorithm (GA) to evolve type stencils (see, e.g., [19] for more details about GAs). The system shares common traits of the works presented in [11] and uses a feasible-unfeasible strategy presented in [16]. We also use an archive to save the evolved individuals based on a similarity criterion [16, 20, 21].

The system is schematically overviewed in Figure 3 and behaves as follows. The evolutionary process begins with the initialisation of the population with randomly created stencils. The fitness of each stencil is calculated according to a fitness function designed by the user. With the stencils evaluated, a new generation of stencils is created using an elitism strategy, i.e. a preset number of fittest stencils proceed unchanged. This step has no effect on the first generation. After the population is evaluated, stencils above a preset threshold are considered feasible stencils. The feasible stencils are compared with the stencils on the archive and if they are dissimilar from the existing stencils they are added to the archive. A stop criterion is tested to determine whether evolution proceeds or stops. If evolution continues, the system determines based on the number of feasible stencils whether novelty search is performed or not, determining how stencils are selected as parents. If novelty search is not performed, stencils are selected by tournament based on their fitness. If novelty search is performed the tournament is based on the fitness and novelty of the individuals of the population and the individuals that are on the archive. Variation operators, *i.e.* crossover and mutation, are applied to the stencils selected as parents to generate offspring stencils. The offspring stencils are evaluated and a new generation is again formed. The whole evolutionary process is repeated until the stop criterion is satisfied. The following subsections detail some mechanisms of the system.

2.1. Representation. Each stencil being evolved consists in a composition of line segments and Bézier curves with varying thicknesses. Therefore, each gene in the genotype of each stencil encodes one line segment or curve in a two-dimensional space.

We implemented an encoding that enables the representation of line segments and Bézier curves. Each gene consists in a 9-tuple with the coordinates of the two endpoints, the angles of the two control points, the lengths of the two control points, and the thickness value. Genes with angle and/or length of the control points set to zero represent straight lines. Figure 4 illustrates the different attributes encoded in each gene: (X1, Y1, X2, Y2, A1, A2, L1, L2, T). The position of the endpoints is constrained by a square grid with a preset density. Also, note that the number of lines may vary from stencil to stencil.

The mapping mechanism that expresses each genotype into its phenotype consists in the drawing of black lines encoded in the genotype on a white canvas. However, the mapping process of the stencils being evolved is not direct. We need one mapping for each character we want to draw with the stencil. This way, we developed a mapping mechanism based on binary masks that define how a given stencil is used to draw a given glyph. When we say *how* we mean which lines are used. This mechanism is illustrated in Figure 5. In what concerns representation, a stencil-based approach may hinder the evolutionary process, because the genetic algorithm has to find a structure of lines (stencil) that is capable of drawing any letter. One could say that we are dealing with a compression problem, i.e. compressing all letters into a stencil. Nonetheless, we believe the "compression" nature behind this stencil-based approach may promote coherence and unity among the resulting glyphs. Furthermore, it helps to understand the anatomy of glyphs and how they share their components/parts. Also, this representation enables us to use an evolved stencil to draw more visual elements other than letters, e.g. signage or symbols, that would be coherent and have the same style as the letters encoded in that stencil.

2.2. Variation. For the initial population the stencils are created at random. We perform variation operations on the stencils using crossover and mutation. The crossover operation consists in the exchange of lines between two stencils. The crossover operator can be summarised to the following steps: (i) select a random rectangular area of the grid; (ii) determine for both parents the lines whose middle points are inside the random rectangular area; and (iii) exchange those lines between the parents. This crossover may be asymmetric as the number of genes it moves from the individual A to individual B may be different from the number of genes it moves from B to A. This results in stencils with a different number of elements in comparison with their parents.

The mutation of a stencil consists in the random modification of genes (lines) of its genotype and comprises three procedures: deletion, modification, and insertion. Each mutation procedure can occur independently with preset probabilities. The deletion procedure selects a line at random and removes it from the stencil. The modification procedure changes one or more lines of the stencil by performing one of the following options, each with a preset probability: (i) moving one of the endpoints by the minimum translation in the grid in one of the eight possible directions; (ii) varying

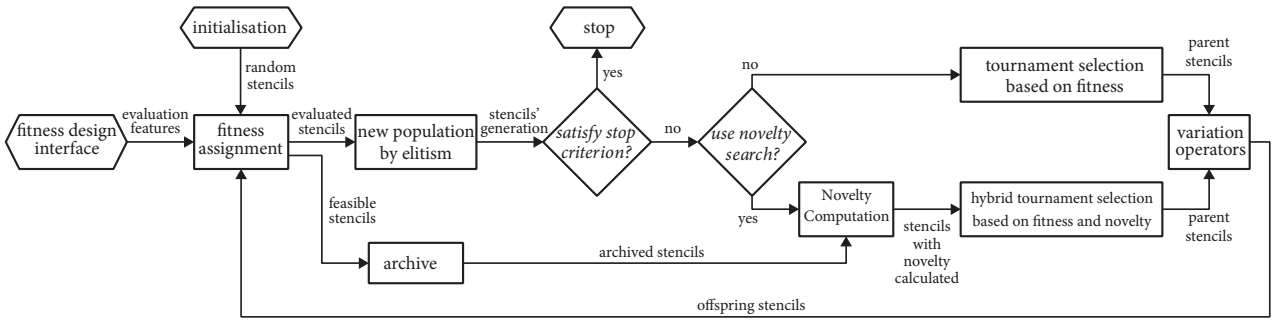

Figure 3: Framework overview.

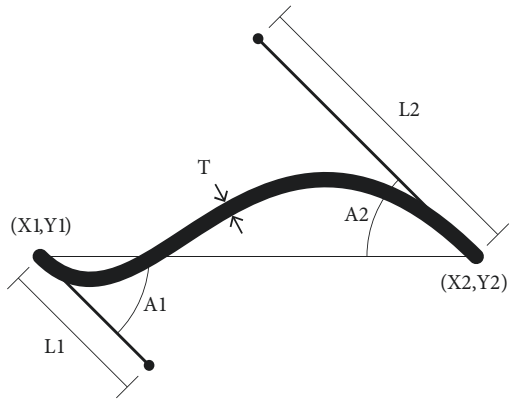

Figure 4: Line encoding.

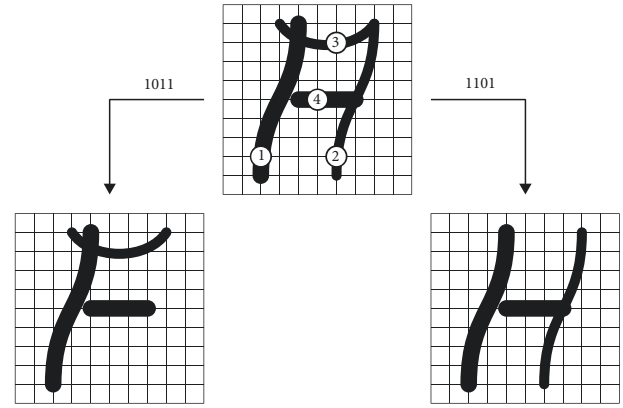

Figure 5: Mapping mechanism expressing the genotype of a stencil (top) into two glyphs (bottom). Binary masks are used to indicate which lines of the stencil should be used to draw the glyphs.

the angle of one of the control points; (iii) varying the length of one of the control points; or (iv) varying the thickness value. Finally, the insertion procedure inserts a new randomly generated line into the stencil. The deletion and insertion procedures cause the variation of the number of lines, enabling the evolution of stencils with different size. Either variation operators preserve the validity of the stencils. A stencil is considered valid if: (i) all its lines are different; (ii) all lines are located inside the limits of the grid; (iii) the number of lines remains within a preset range; (iv) no line has null length; and (v) no line contains another one.

2.3. Evaluation. Based on the work of Romero et al. [22] and previous approaches [9, 20, 23, 24], we adopt an automatic fitness assignment scheme to evaluate the individuals, i.e. stencils, and this way autonomously guide the evolutionary process.

The evaluation of each stencil consists in the computation of (i) behaviour features, related to how the stencil performs in drawing glyphs for the target characters, and (ii) morphology features, related to its structure and components. We conceived the fitness assignment to enable any combination of features to be pursued by the evolutionary process. Furthermore, in a combination of features, each feature can have more or less importance (weight) in comparison to the other features. As a result, the fitness function consists in a weighted product:

$$fitness\ (ind) = \prod_{i}^{n} \left(satisfaction_i\,(ind) * w_i + 1 - w_i\right), \quad (1)$$

$$satisfaction\,(ind) = 1 - \left|featurevalue_i\,(ind) - t_i\right|, \quad (2)$$

with $wi, vi, ti\ [0,1]$, where w_i is the weight of the ith feature, t_i is the desired target value for the feature i, and $featurevalue_i$ is the value measured corresponding to the feature i. The evolutionary process aims at maximising the value of Equation (1), whose theoretical optimum corresponds to a stencil that simultaneously matches all features to their target values according to Equation (2). All the features and weights are normalised to the [0..1] domain. As such, maximising or minimising a given feature consists in setting its target value to 1 or 0, respectively.

The following subsections detail the behaviour and morphology features of the stencils.

2.3.1. Behaviour. One of the preconditions of the stencils evolved with the proposed framework is their ability to produce glyphs that are legible, *i.e.* stencils should be able

Table 1: Behaviour features.

feature	description
recognisability	character recognition using a Convolutional Neural Network (confidence value of the classifier in recognising the potential glyph as the target character)
similarity	visual similarity to existing glyphs using Self-Organising Maps (RMSE pixel-by-pixel similarity between the potential glyph and the most similar neuron in the Self-Organising Map of the target character)

to express images that are recognised as characters. The framework evaluates this ability by measuring features based on the glyphs that each stencil is able to express. Table 1 presents an overview of these features, which we refer to as behaviour features.

The expression of each stencil into glyphs takes a couple of steps. As explained before, each stencil has several lines that can be activated to draw glyphs. First, the system chooses a character for which the stencil has to produce glyphs. Next, among the lines that compose the stencil, we search which mask of active lines better expresses glyphs for the chosen character. This way, each mask stores the best use, or configuration, of the stencil found during the evaluation process to draw a given character.

We use a hill-climbing algorithm to perform the search for the best configuration of the stencil being evaluated for each target character. This way, the evolutionary process includes a nested search to find optimal configurations for each stencil. The search starts with all the lines deactivated, activating one per step. At each step, all newly generated configurations are evaluated as a glyph for the target character, *i.e.*, how the expressed glyph matches the target similarity and recognisability values. The search stops when no improvement in the evaluation is achieved, storing the best mask and evaluation value of the resulting glyph. This process is repeated for all the target characters. In a typical evolution, all target characters are equally considered, i.e. the stencils being evolved should be able to draw glyphs for all of them. However, the user can specify different importance levels for the target characters. This enables the user, for example, to evolve stencils that can express glyphs for a subset of characters, or to improve the legibility of some glyphs of an evolving stencil.

Recognisability. We use a Convolutional Neural Network (CNN) classifier to calculate how much a glyph is recognised as a given character. CNNs are a type of Deep Neural Networks (DNNs) that have been used successfully in image classification and recognition tasks [25, 26]. The main characteristic of a CNN is the usage of convolutional and pooling layers, which provide feature extraction and dimensionality reduction in training [27]. Each layer can be seen as a filter from which features are extracted and learnt.

The architecture of the CNN is based on the *Lenet*-5 network for digits recognition [28] but trained as multi-class supervised classifier for the 26 capital letters of the Roman alphabet. Our approach must perform several evaluations of several glyphs per generation, therefore the chosen network was a trade-off between computational power and efficiency. The classifier is trained on the 32-by-32 pixel representation of the typefaces served by Google Fonts. Besides the typefaces, we added a negative class represented with random images generated by our approach that do not resemble any characters, yielding a total of 27 classes. The value of the recognisability feature of a stencil configuration is the output of the classifier, indicating its confidence in recognising the input image of the configuration as its target character. An output of 1 indicates total confidence while an output of 0 indicates the opposite. Note that the Machine Learning model used is based on data available and off-the-shelf architectures with its inherited limitations and exploits [29, 30]. Furthermore, the models employed are not able to harness the full potential of the human-like visual system. With this in mind, this feature is used to evaluate the legibility, recognisability and readability of the input images.

Similarity. An array of Self-Organising Maps (SOMs), one for each target character, is used to calculate the similarity feature. The SOM [31, 32] is among the most well-known unsupervised learning and clustering approaches. The architecture of the SOM consists in a feed-forward neural network that reduces information while preserving the most important topological relationships of the data elements. This enables the calculation of the visual similarity of the glyphs expressed by each stencil with existing glyphs.

Each SOM is constructed of 64 neurons and is trained with 32-by-32 pixels images of several glyphs of the corresponding target character. The glyphs used for training were gathered from the typefaces of the Google Fonts platform.

To calculate the similarity feature of a stencil configuration (glyph), an image representation of it is created with the same size as the SOM neurons and then compared with each neuron of the target character SOM. In this comparison, the similarity between the image representation of the stencil configuration and each neuron in the SOM is calculated using the Root Mean Square Error (RMSE), which measures how close, or far, the candidate glyph is to a reference glyph (neuron) on a pixel-by-pixel basis. The value of the similarity feature considers the distance between the glyph expressed by the stencil and the most similar neuron in the SOM, also called the best matching unit, and is given by: $1-\text{normalised}_{\text{RMSE}}$.

2.3.2. Morphology. In addition to the behaviour features, the framework considers a series of other features related to the structure and components of the stencil. Table 2 presents an overview of these features, which we refer to as morphology features.

By adjusting the morphology features, one can promote the evolution of stencils that exhibit particular structural

Table 2: Morphology features.

feature	description
size	number of stencil lines (normalised to the [0..1] range according to a preset range)
coverage	rectangular area occupied by the stencil lines (normalised to the [0..1] range according to the grid area)
continuity	percentage of stencil lines that share endpoints with other line
intersection	percentage of stencil lines that intersect other line
parallelism	percentage of stencil lines that are parallel to other line
horizontal symmetry	similarity between the top half of the stencil and the bottom half mirrored vertically (calculated using the RMSE between the top and bottom half)
vertical symmetry	similarity between the left half of the stencil and the right half mirrored horizontally (calculated using the RMSE between the left and right half)
curves	percentage of stencil lines that are curves
symmetric curves	percentage of stencil curves that are symmetric in relation to the line that (i) is perpendicular to line segment S and (ii) intersects the middle point of S; where S is the line segment that connects the two end points of the curved line.
length	average length of the stencil lines (normalised to the [0..1] range)
length diversity	standard deviation of the lengths of the stencil lines (normalised to the [0..1] range)
thickness	average thickness of the stencil lines (normalised to the [0..1] range)
thickness diversity	standard deviation of the thicknesses for the stencil lines

characteristics. The possibilities are vast. For instance, one can configure the morphology features of the fitness function to reward stencils with only curved lines that intersect little, or horizontally symmetric stencils with only straight lines with great continuity, or stencils with long curved lines that intersect little.

The combination of morphology and behaviour features enables the evolution of stencils that match particular visual characteristics while ensuring the legibility of the glyphs. This way, the user can explore different compromises between the legibility and the style provided by the generated stencils.

2.4. Archive. Similar to the work done in [16], the archive is used to evaluate our solutions during the evolutionary process and prevents the algorithm from searching areas of the search space that were already visited. The archive should hold the set of stencils found to date by the evolutionary process. The size of the archive shows how the algorithm can generate diversified stencils.

The archive comes into play after the fitness assignment. At this stage, a candidate stencil has its fitness assigned, and it has to meet two requirements in order to be added to the archive: (i) its fitness must be greater than or equal to an adequacy threshold f_{min}; (ii) it needs to surpass a dissimilarity threshold when compared to those that already belong to the archive.

This process is performed by computing the average dissimilarity between the candidate and a set of k-nearest neighbours. When the average dissimilarity is above a predefined dissimilarity threshold, $dissim_{min}$, the individual is added to the archive. In this approach, we evaluate the stencils based on their expression as glyphs, i. e., the stencils are analysed in the form of images of their glyphs. As in [16], the dissimilarity metric for an image i is computed as:

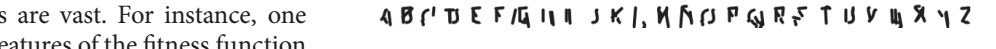

Figure 6: A stencil's expression rendered to a single image, which is used to compute the similarity between stencils for the archive algorithm.

$$dissim(i) = \frac{1}{max_{arch}} \sum_{j}^{max_{arch}} d(i, j) \quad (3)$$

Where max_{arch} is a predefined parameter for the number of most similar images to consider when comparing with image i and $d(i, j)$ is a distance metric. The distance metric measures how different two images are. There are two exceptions to the application of this measure: (i) if the archive is empty then the first stencil that has a fitness above the f_{min} is added; and (ii) if the number of entries on the archive is less than max_{arch} we use the number of existing entries instead of max_{arch}.

In order to evaluate the similarity between stencils, we resort to an image similarity metric applied to the stencil's behaviour, i.e. the image output of the configurations for each letter. One image is created containing several letters concatenated to form a "banner" image as shown in Figure 6.

The banner is used to evaluate the similarity among the several candidate stencils and the archived ones. We use a similarity metric the RMSE between the pixels. It is out of the scope of this work to explore several dissimilarity metrics. Since we are processing a considerable number of images per generation, we resorted to RMSE for its fast calculation. To the interested reader, we suggest consulting the works by [33, 34] for more detail on similarity and dissimilarity metrics. When a stencil is added to the archive, it counts as a feasible solution.

The mechanism that selects feasible solutions is important to shape how evolution will proceed, depending on the

results obtained in a given generation. We introduce the novelty approach presented in [16], a customised selection mechanism that can switch between a fitness-based strategy and a hybrid mechanism that considers both fitness and novelty. As depicted in Figure 3 it can switch between fitness and hybrid according to the following decision rule: if the number of feasible solutions of the current generation is lower than a threshold T_{min} change to fitness guided evolution; or if the number of feasible solutions of the current generation is above a threshold T_{max} change to hybrid mechanism. In fitness guided evolution, the tournament selection is based on the fitness values of the candidate solutions, as in a standard Evolutionary Algorithm (EA). If hybrid is chosen, it is necessary to compute the novelty of each selected individual, and perform a Pareto-based tournament selection, using the novelty and fitness of each selected individual as two different objectives to maximise. The novelty computation process is inspired by Lehman and Stanley's work [35], with one small change: the k most similar images are considered from the set of the selected individuals and the archive, instead of considering the whole population and the archive. At this stage, each selected individual has a fitness and novelty value, and there is the need to determine the winner of the tournament. This process is inspired by multi-objective EAs, namely the Pareto-based approaches, which select the best individuals based on their dominance or non-dominance when compared to other individuals. In this work, the hybrid tournament selection determines the non-dominant solutions by comparing, among the selected individuals, both fitness and novelty. After the set of non-dominant individuals are computed, we have the so-called Pareto front. Using the hybrid mechanism, the tournament winner is selected by randomly retrieving one of the solutions of the Pareto front.

2.5. Implementation. The proposed framework is implemented in two modules: (i) the evolutionary system and (ii) the fitness function design interface. The first module, the evolutionary system, operates the GA and the fitness assignment scheme that automatically guides it. The second module, the fitness function design interface, enables the user to adjust parameters of the fitness assignment and other inner workings of the first module. Figure 7 shows a screenshot of the evolutionary system (left) and the fitness function design interface (right).

The fitness function design interface communicates with the evolutionary system through a JSON file. Technically, one could manually adjust the parameters stored in that file and this way use the evolutionary system alone to evolve stencils. However, this would hinder the design of fitness functions and the configuration of the evolutionary process.

A typical use of the framework could be initiated as follows. The user launches the evolutionary system and selects the source of the setup parameters: (i) from a setup file or (ii) from the fitness function design interface. When the first source is selected, the user imports a setup file stored in the computer, *e.g.*, a setup file previously exported using the fitness function design interface. This approach is useful to test a series of experimental setups. When the second source is selected, the evolutionary system activates a mode in which it listens for new parameters coming from the fitness function design interface. In this approach, the user uses the fitness function design interface to adjust the setup parameters, which are directly sent to the evolutionary system. This enables the user, for example, to modify the fitness function during the evolutionary process. After selecting the source of the setup parameters, the user is in position to evolve stencils. In the following subsections, we overview some of the key functionalities of the two modules.

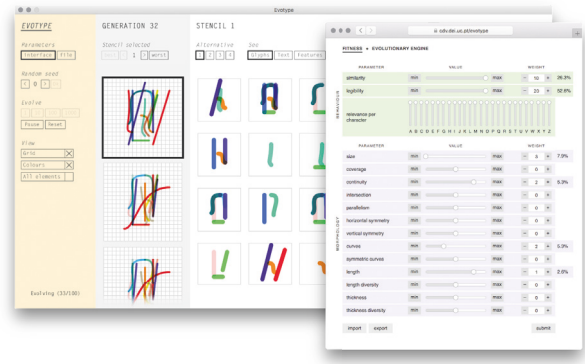

Figure 7: Screenshot of the framework, consisting of the evolutionary system (back) and the fitness function design interface (front). A demo video can be seen at http://cdv.dei.uc.pt/2018/complexity/evotype.mov.

2.5.1. Evolutionary System. The evolutionary system module provides the necessary means to evolve, browse, test, and export type stencils. After selecting the setup parameters, one can command the evolution of stencils by setting the random seed and instructing the system to generate a given number of generations. During evolution, it is possible to browse throughout the current generation of stencils, which are arranged vertically by descending order of fitness. The system features a mode that renders the elements of each stencil using different colours, either when previewing the entire stencil or the glyphs produced with it. The purpose of this mode is to visualise the reuse of elements of the stencil between the different glyphs. The user can select each stencil to (i) test it by typing a couple of words with the glyphs produced with it; (ii) visualise its features, or measurements, that are being considered by the fitness function; and (iii) export it to file, enabling further refinements and its utilisation outside the framework.

2.5.2. Fitness Function Design Interface. The module of the fitness function design interface enables one to design fitness functions to automatically guide the evolution of type stencils. The interface empowers the user by enabling him/her to express preferences through the specification of properties that he/she intends to observe in the evolved stencils. Although the main goal of the interface is the configuration of the fitness function, it also enables the adjustment of several

parameters of the evolutionary process, *e.g.* population size, elite size, tournament size, crossover rate, mutation rate, grid size, phenotype size, and other parameters related to the novelty search approach, including the minimum fitness for a stencil to be considered feasible and the minimum dissimilarity to be added to the archive.

The fitness function design interface consists of a web page with multiple sliders and buttons that enable one to adjust evaluation and evolution parameters in a high-level way. One could say this module acts as an interactive facilitator of parameterisation of the first one, the evolutionary system, abstracting the user from the inner workings of the framework. This approach enables the user to submit parameters to the evolutionary system at any time, as already explained, and this way develop or change his/her preferences throughout the generations. In addition to submitting the current parameters to the evolutionary system, the user can also export the parameters to file and import them later. The decision of implementing this module as a web page is related to our short-term goal of converting the framework into a web application. This would enable anyone to use the framework.

Based on the two levels of evaluation identified at the beginning of this section, the fitness parameters presented in the interface were arranged into two groups that are visually distinguished using different colours. The top group of parameters is related to the behaviour of the stencil, while the bottom group is related to its morphology. The interface employs tooltips to enable the user to understand the different components of the interface, *e.g.* the semantics associated with each feature and how it is calculated. When the user hovers the cursor over a component, a tooltip appears displaying information about it.

For each parameter, the user may set (i) the value that should be matched by the stencils being evolved and (ii) a weight that indicates the importance of that parameter in the fitness function. The only exception is the last parameter of the behaviour group, which presents an array of vertical sliders to specify the relevance of each character the evolved stencils should be able to draw glyphs for. Changing the weight of one parameter results in having more or less impact in comparison to the other parameters. In order to make the adjustment of weights easier to understand, we adopted an approach where the user indicates each weight by adding or subtracting units to the weight. Nonetheless, one can also set the weight to a specific floating value. For instance, one parameter with a weight of 3 would have an importance three times greater than a parameter with a weight of 1. Following this reasoning, setting the weight of one parameter to 0 means that it will be ignored. One advantage of this approach is the precision it provides to the user when adjusting each weight, in comparison to other approaches that employ, for example, sliders. The final weight of each parameter, considering the other weights, is displayed on the right side. This information helps the user understanding the overall impact of each individual parameter in the fitness function. Also, by only displaying the final weights greater than zero, we are able to visually highlight the parameters that are being considered.

3. Experimentation

We conduct four experiments on the proposed framework with different goals in mind. In the first experiment, we study the adequacy of the hybrid fitness function (similarity and recognisability) for guiding the evolutionary process. In the second experiment, we analyse how the design of fitness functions influences the outcomes of the system and if, and to what extent, it is able to convey the preferences of the user. In the third experiment, we investigate the impact of novelty search on the evolutionary convergence and on the diversity of stencils evolved. In the last experiment, we explore the creative possibilities provided by the outputs of the presented framework by using one evolved stencil in a design project.

In this work, we evolve stencils to draw glyphs for the uppercase letters of the Roman alphabet. The base experimental parameters are summarised in Table 3.

3.1. Experiment I - Hardwired Fitness Functions. In this section, we analyse the ability of the approach to evolve stencils with hardwired fitness functions. In [11] we validated that the evolutionary engine by performing experiments that used and hardwired fitness function resorting to RMSE for a predetermined target typeface. The results have shown that the evolutionary algorithm is able to evolve stencils that expressed visually coherent glyphs. However, in the first set of experiments, the evolutionary algorithm generated stencils which maximised the number of elements and some presented several gaps. In the second set of experiments, we redefined the fitness function to control the number of elements and minimize gaps. The approach was able to generate simpler stencils able to produce glyphs similar to the targets using lesser elements, promoting the reuse of stencil's elements for multiple glyphs. An overall observation is that in order to promote a specific behaviour we have to redefine the hardwired fitness function, which is a sensible and time-consuming process.

In the experiments of [11] the only behaviour feature used to guide the fitness was the similarity feature. It was based on the pixel-based RMSE between a glyph expression to a predefined target typeface glyph. In order to promote more flexibility in the solutions, we use a SOM for the calculation of the similarity feature. The SOM organises and reduces the instance space. We use RMSE to compute the similarity of a candidate glyph expression with an expression of the closest SOM neuron. The SOM trained with several typefaces provides different targets to explore while reducing the number of targets to be evaluated. This allows for a more flexible evaluation of similarity. Nevertheless, there is a possibility of the approach exploring the activations of different SOM neurons belonging to different glyphs. We also introduce the concept of recognisability, performed by a CNN. The idea of using the CNN evaluation as part of the concept of recognisability is to promote the generation of stencils which retain recognisable characteristics of existing glyphs.

In Table 4 we present three fitness functions defined with different recognisability and similarity weights. The size and continuity features were maintained from the experiments

Table 3: Experimental Parameters.

parameter	value
generations	1000
population size	100
elite size	1
selection	Tournament
tournament size	3
rate crossover	0.5
max genes	40
rate deletion	0.05
rate insertion	0.05
rate modification	1 / genotype size
grid size	20 x 20
min length permitted (value relative to grid size)	0.15
control points angles permitted (rotation angles in degrees and relative to the line segment that connects the endpoints)	[−90, −45, 0, 45, 90]
control points lengths permitted (values relative to the distance between the endpoints)	[0.25, 0.5, 0.75]
thickness values permitted (values relative to the phenotype size)	[0.125]
phenotype size	32 x 32
hill-climbers	1

Table 4: Fitness functions designed and tested in experiment I.

fitness function	feature	target value	weight
fitRec	recognisability	1	100
	similarity	-	0
	size	0	2
	continuity	1	2
fitSim	recognisability	0	0
	similarity	1	100
	size	0	2
	continuity	1	2
fitHybrid	recognisability	1	67
	similarity	1	33
	size	0	2
	continuity	1	2

in [11]. In all the experiments we track the response values of each feature that compose the final fitness function for analysis purposes, even if the weight is set 0, i.e. it does not participate in the calculation of the fitness.

In *fitRec*, the evolutionary process is mainly guided by the recognisability feature, i.e., based on the activation of the CNN for each stencil's glyph expression. In Figure 8, on the top left plot we can observe the behaviour of the evolutionary algorithm using the *fitRec* fitness function, showing that we are able to guide evolution and optimise the stencils' fitness. It starts with a relatively low fitness value but rapidly converges to high fitness values in a few generations. If we analyse the progression of the values of the other components, we can observe that the similarity is not affected by the progression of the recognisability. The size feature in the first generations tends to rapidly increase, meaning that elements are being removed from the stencil. When the fitness stabilises, the size feature increases, expressing the highest value amongst the tested functions. It means that in the end, it has fewer elements than the other fitness functions. Regarding continuity, it consistently rises along the generations. Based on the values at the end of the evolutionary process, it seems to create disconnected stencils when compared with the other fitness results.

The *fitSim* function uses the similarity feature to guide fitness. As shown in Figure 8, the evolutionary algorithm is able to optimise the fitness function, although it does not reach the maximum theoretical value. The recognisability values tend to increase with the increase of the similarity feature values. The size component consistently increases, suggesting that the best stencil tends to remove elements along the generations. When compared to the others, *fitSim* reaches to the highest number of elements used by the stencil. The high values of the continuity feature show that it tends to create a connected stencil.

The *fitHybrid* is a fitness function that combines both the similarity and the recognisability to guide evolution. The values of Table 4 show that more weight was given towards the recognisability. The idea is to have stencils able to produce expressive and functional typefaces, exploring different compromises between the expressiveness and the legibility of the glyphs while maintaining coherence. Once again, we

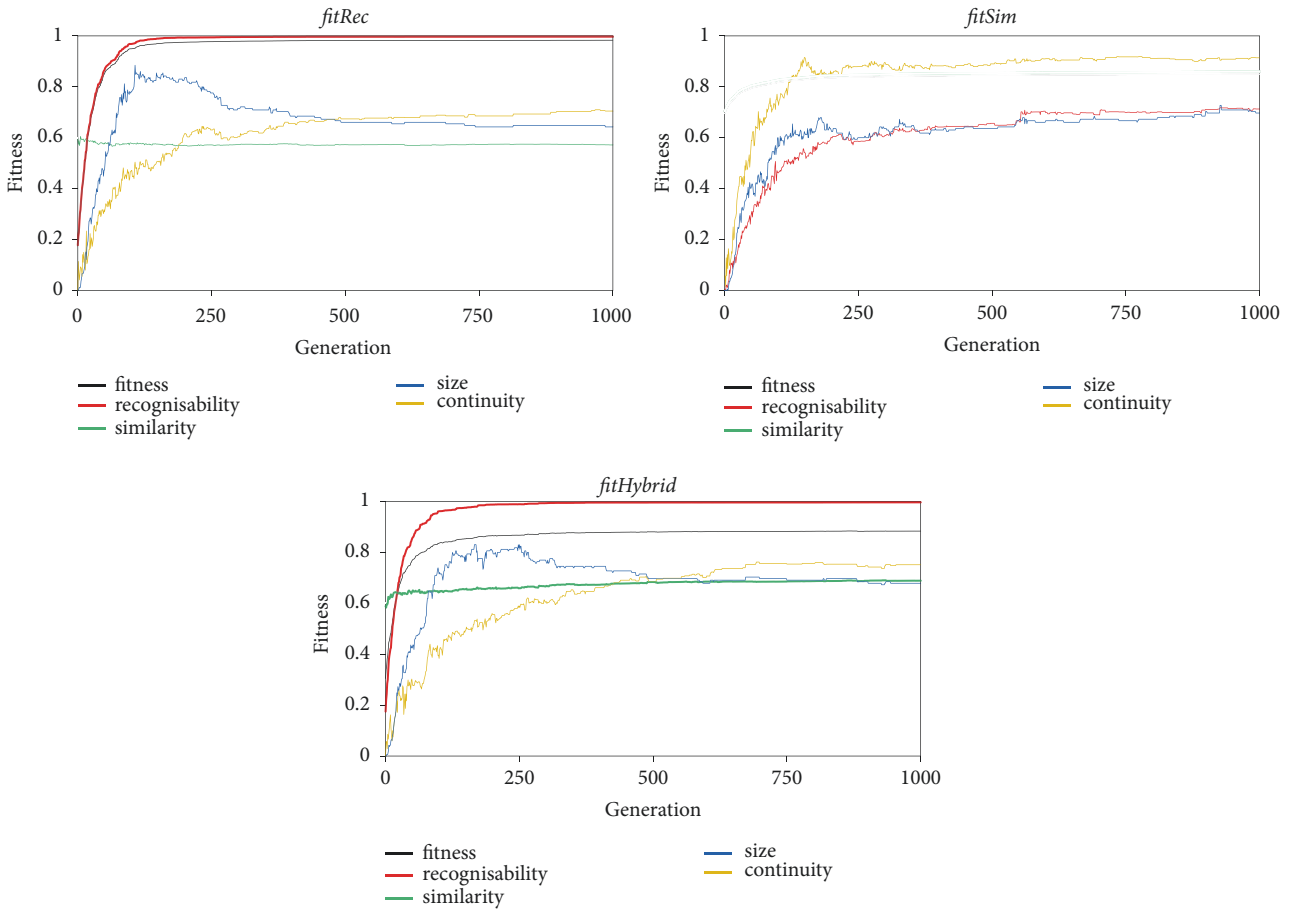

FIGURE 8: Progression of the fitness and features' values of the fittest stencil when fitness functions *fitRec* (top left plot), *fitSim* (top right plot), and *fitHybrid* (bottom plot) guide the evolutionary process. The visualised results are the average of 30 runs.

are able to guide the evolutionary process and optimise the fitness function. In terms of fitness, it is possible to observe that it maintains a certain stable level of similarity and that the recognisability contributes more to the fitness increase. In terms of the other two features, size and continuity, we can say that we get the good from both fitness features, i.e., a low number of elements and a more connected stencil.

In Figure 9 we can observe generated stencils for the fitness function used in this first experiment. It is noticeable the difference between them at the visual level. The results also cope with the analysis in terms of fitness components. *fitSim* tends to generate stencils with more and connected elements. Although the SOM gives us more flexibility than the RMSE target approach of [11], in this approach it generates stencils that fill the space of the target neurons of the SOM. The flexibility comes with a trade-off, some of the glyphs generated by the stencil appear to focus on different SOM neurons, resulting in different glyphs expressions. *fitRec* uses fewer elements but the used elements are more disconnected and dispersed. However, we see some random features around the generated glyphs that can be seen as exploits of the classifier guiding the evolution. We are aware of the propensity of the evolutionary algorithms to find shortcuts and exploit weaknesses on fitness assignment schemes that use ML approaches [28–30]. In *fitHybrid* we combined the recognisability with the similarity to prevent the approach to guide the evolution to recognisable but atypical glyphs. *fitHybrid* tends to generate stencils with a small number of elements that are connected generating glyphs that are simpler, distinguishable elements, demonstrating variability while maintaining coherence. Overall, we consider that using the fitness functions defined for this experiment we are still able to evolve stencils that generate coherent glyphs.

In the previous set of experiments, we used a stricter evaluation based on a target font [11]. The algorithm converged to a structural representation of that font, i.e. a stencil to draw it, showing that the approach can arguably generate a compressed representation of a font. In the experiments presented in this section, we observe a similar behaviour. This enforces the idea that the representation is adequate to the task at hand.

The results show that we can evolve recognisable and legible fonts, but this is not enough for them to be aesthetically appealing. Performing more generations could marginally augment the aesthetics of the results but would not lead us to the high-quality solutions of a commercial font. This fact leads us to two different hypotheses not mutually exclusive about type design. When a type designer creates a font, it does

Figure 9: Typical results evolved in different runs of *fitRec* (top group), *fitSim* (middle group) and *fitHybrid* (bottom group). To better identify each element of the stencils (left) in the corresponding glyphs (right), a random colour is used for each element.

not look exclusively into its functionality [31–33]. The visual features and ML approaches in use are not able to harness the potential of the human-like visual system to guarantee that if something is legible from the machine point of view is legible for the human and that the factors that lead to an increase or decrease in terms of legibility, readability and recognisability to a human are the same for the machine and vice-versa. Assuming that this is the case, if we use more complex visual models, trained to recognise other types of artefacts and, as such, subject to the tasks that a human is subject to deal in a daily basis it could contribute to enhancing the results.

Based on overall results and discussion, we consider that experimenting with fitness function design in a semi-automatic way can be beneficial. This path is explored in the next set of experiments.

3.2. Experiment II – Designing Fitness Functions. In this experiment, we analyse how the design of fitness functions influence the outcome of the framework and if, and to what extent, they are able to convey the specified preferences. We focus on the morphology features because these are likely to be perceived visually on the stencil as well as on the resulting glyphs.

Using the experimental setup tested in experiment I as a base, we add other features to the fitness function. We design and test 4 more fitness functions. Each one consists in the base fitness function (*fitHybrid*) combined with one, or two, more morphology feature(s). The features added to *fitHybrid*, along with a name for the resulting fitness function, are listed in Table 5.

Although many different combinations of features could be tested, we selected some that we believe can be more noticeable in the evolved stencils. Since the encoding of Bézier lines is an iteration of this framework (comparing to our previous work [11]), we also focused this experiment on features related to them.

Table 5: Fitness functions designed and tested in experiment II.

fitness function	feature	target value	weight
fitCurves	curves	1	4
fitNoCurves	curves	0	4
fitSymCurves	curves	0.5	4
	symmetric curves	1	4
fitUniformLength	curves	0.5	2
	length	0.66	4

We designed each fitness function to set the framework to evolve stencils with specific visual characteristics:

(i) *fitCurves* — stencils entirely composed of curves;

(ii) *fitNoCurves* — stencils with no curves;

(iii) *fitSymCurves* — stencils with half of their elements being symmetrical curves;

(iv) *fitUniformLength* — stencils also with half of their elements being curves, and all lines should have a length of two-thirds of the grid size.

Figure 10 summarises the results of this experiment. Per fitness function, we visualise the progression of each feature being evaluated and present one typical stencil evolved using that fitness function. In general, and based on the different runs of each fitness function, the results indicate that: (i) different fitness functions lead to different stencils; (ii) different runs with the same fitness function converge to different stencils, thus providing diverse stencils; and (iii) the four fitness functions designed are able to guide evolution towards stencils with features that match the preferences behind them, sometimes in surprising ways.

The framework frequently finds interesting ways to match the fitness functions, generating unusual glyphs more or less functional. On the other hand, sometimes, the framework generates stencils that, from the type design point of view, may be appealing due to their novelty and aesthetics, but which do not maximise all features of the fitness function. Nevertheless, the results reveal the effectiveness of the approach in exploring possibilities that are consistent with the preferences expressed by the user who designed the fitness function. Looking at each stencil, and their glyphs, in Figure 10, one can see that they exhibit visual properties that are aligned with the features added to each fitness function. For instance, the stencil evolved with *fitCurves* is almost only composed of curves (only one line segment is used); on the other hand, the stencil evolved with *fitNoCurves* is almost only composed of line segments (only three curves are used); the stencil evolved with *fitSymCurves*, in addition to using the same number of curves of line segments, most of the curves used are symmetric (only one curve is asymmetric); and the stencil evolved with *fitUniformLength*, has the same balance of curves and line segments as the previous stencil all

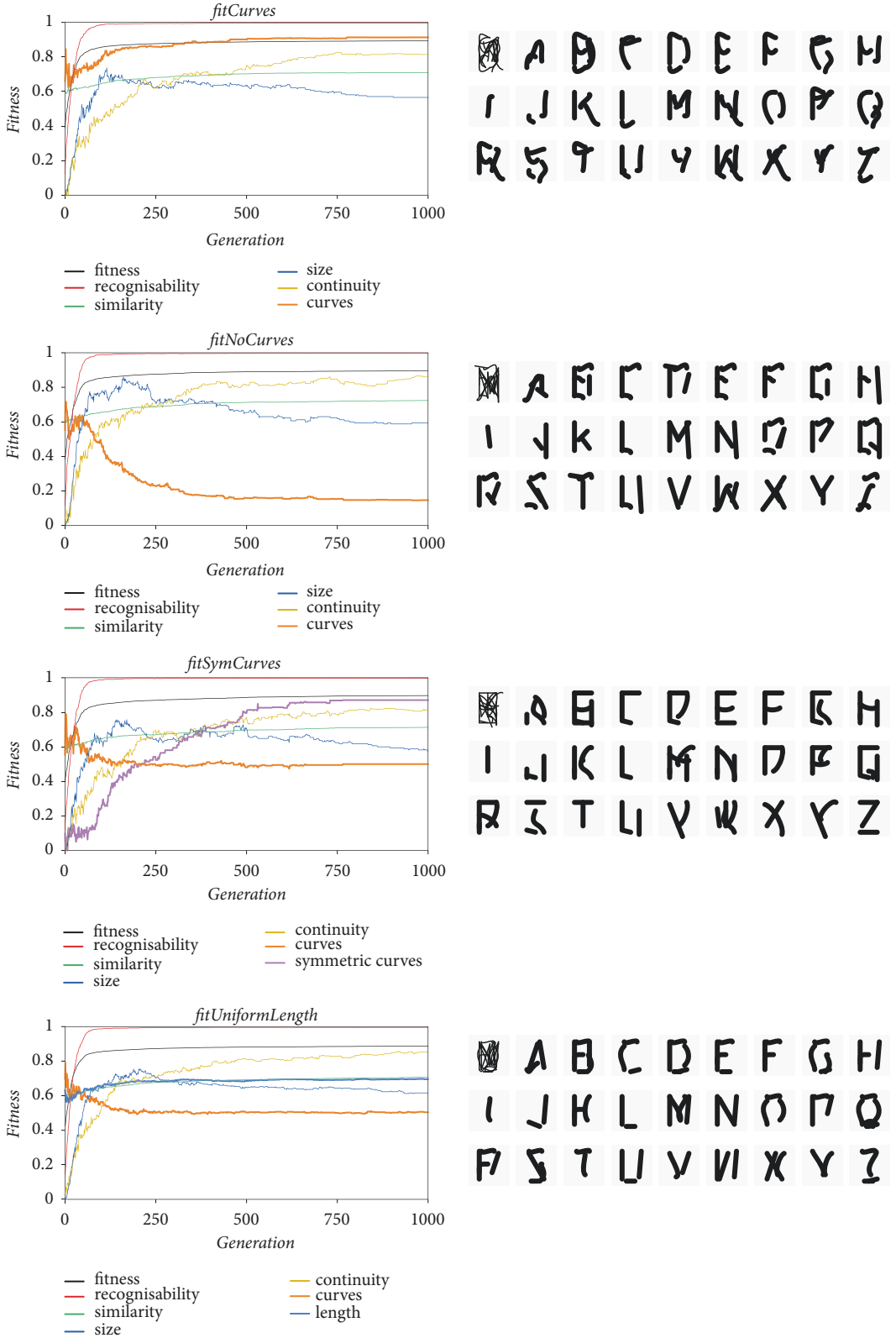

Figure 10: Experimental results when the evolutionary process is guided by *fitCurves*, *fitNoCurves*, *fitSymCurves*, and *fitUniformLength*, in descending order. For each fitness function, one can see the progression of the fitness and features' values of the fittest stencil over the generations (left) and one typical fittest stencil of the last generation (right). The visualised results are the average of 10 runs.

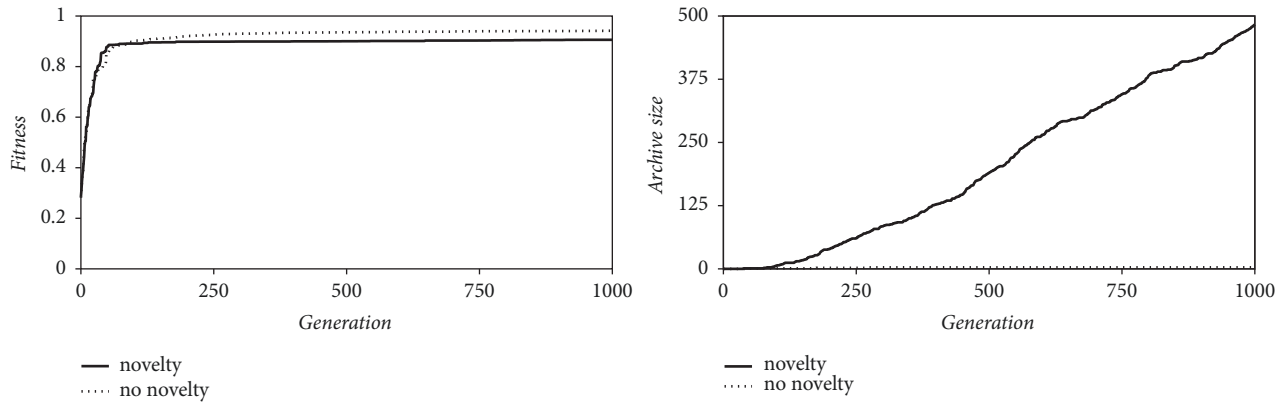

Figure 11: On the left, the progression of the fitness and features' values of the fittest stencil across generations using the *novelty* and *no novelty* setups. On the right, the evolution of the archive size across generations. The visualised results are from a single run.

Table 6: Experimental parameters.

parameter	value
t_{min}	5
t_{max}	15
max_{arch}	5
f_{min}	0.85
Dissimilarity metric	RMSE
$Dissim_{min}$	0.66

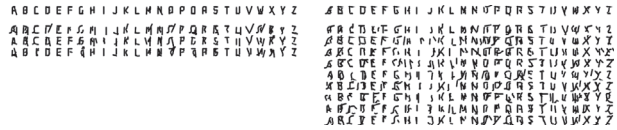

Figure 12: On the left, the archive (sampled) of the expressions from a run of evolving stencils guided by the hybrid mechanism. On the right, the archive (sampled) of a run evolving stencils guided by fitness. The top row instances are the fittest stencil found and the remaining ones above them are the archive for the *no novelty* (on the left) and *novelty* (on the right).

elements have more or less the same length, approximately two thirds the size of the stencil grid.

Overall, evolution was able to optimise all features being tested without compromising the behaviour features, i.e. similarity and recognisability of the glyphs. This should be related to the substantial differences between the high weights used for the behaviour features and the low weights used for the morphology features.

3.3. Experiment III – Novelty Search Mechanisms. In this experiment we assess the ability of the novelty search mechanism to generate diverse stencils in a single evolutionary run and the adequacy of the archive, produced during the process, to summarise the results that are presented to the user. We preserve the experimental setup of *fitHybrid* from the experiment I and used the parameters in Table 6 to be used by the novelty mechanism.

For this experiment, we perform a single run of the following setups: *novelty* - uses the novelty search mechanism; *no novelty* - does not use novelty search, the fitness guided approach. The archive is used on both runs to analyse the impact of the novelty mechanism.

In terms of fitness, in both cases we have a behaviour similar to the one observed in experiment I, it optimizes the fitness function. However, it is noticeable the differences between the two setups in terms of fitness values along the generations. In generation 50 we have the first entry to the archive on both setups, i.e. at least one individual that surpasses the f_{min} value. A few generations after that point, the novelty setup enters in hybrid tournament and we can observe that it rapidly increases up to a certain point, surpassing even the values observed for the fitness guided in the same interval, for a few generations. Around the 100^{th}, generation the *no novelty* setup surpasses the maximum value and continues increasing for a few generations until it stabilizes. We can observe that for the *novelty* setup it continues to slowly increase until the last generation. If we observe in Figure 11 the archive size, we can see clearly that the novelty setup adds much more instances to the archive, suggesting that adds a lot of diversity to the evolutionary process.

Figure 12 shows samples from the archive and the fittest stencils found in *no novelty* and *novelty* setups. We start by analysing the fittest stencils (top left and top right images of Figure 12), observing that they are different from each other which suggests that using novelty impacts the final result of the evolutionary process. Moving towards the analysis of the archives, on the left we show that the *no novelty* setup only adds three stencils to the archive, indicating that for the defined parameterization it does not found any stencil behaviour more dissimilar than the three that we observe in Figure 12. There is some dissimilarity among the stencils but some of the letters remain very similar, e.g., "I", "J", "l" and "Z". We can see that the archive stencils' letters have some resemblance with the fittest stencil's letters. In novelty setup, while analysing the values in the archive size, we see that a lot of different stencils are added to the archive. Due

to the high number of entries on the novelty setup archive we employ a filter using RMSE to select the top-10 most dissimilar instances on the archive. We clearly have some diversity amongst the most dissimilar entries. Some of them contain a mixture of clear letters with some letters that from a subjective standpoint do not resemble the corresponding letter (e.g. Figure 12 on the right in the 6th, 9th, 10th rows). Overall, we can see that the evolutionary process explores a larger area of the search space when compared with the *no novelty* setup. Note that since we are only performing a single run, we can say that is possible to create more diversity and generate a more diverse archive of solutions when compared to the traditional fitness guided solution (*no novelty*). The trade-offs are in the extra parameterization that can be difficult to tune, and it could come at the cost of later convergence. Aside from the extra fine tuning, this suffers from the same problem of tuning hardwired functions, since the user is only a spectator on this process once the evolutionary process starts.

3.4. Experiment IV – Applying Evolved Stencils. In this last experiment, stencils evolved with the presented framework are applied in a real design project: an interactive installation integrated in a permanent exhibition dedicated to Portuguese literature that enables visitors of a museum to generate their own portraits made of letters.

The creation of imagery using text is a traditional design task and is nothing but new. The process of drawing with text can be traced back to manuscripts from many centuries ago with illustrations made of handwritten words. Fast forward to the late 1890s, Typewriter Art becomes an art form, with the first piece of known Typewriter Art being documented, an image of a butterfly created by Flora Stacey in 1898 [36]. Later, in 1966, at Bell Laboratories, Kenneth Knowlton and Leon Harmon created the image "Studies in Perception #1", one of the earliest known examples of ASCII art and probably the first computer nude. To create the image, Knowlton and Harmon scanned a photograph and assigned typographic symbols to the binary numbers according to halftone densities [37].

The interactive installation employs a generative process based on ASCII art to create the portraits. It dynamically changes the typographic weight of each letter to render different shades of grey and this way depict an input image.

The mapping of the darkness of the input image into letters is best accomplished using a typeface with many weights, so a continuous range of shades of grey can be rendered typographically. We believe stencils evolved by the presented framework can play a role here because by using a stencil, each letter can be drawn with as many thickness values as needed. In other words, a continuous range of thickness values can be used to give body to the letters in the portrait and this way enable the representation of different shades.

The video at http://cdv.dei.uc.pt/2018/complexity/evotype.mov shows the interaction with the framework in order to design fitness functions to guide an evolutionary run and this way evolve a series of stencils. After selecting the stencils from the framework archive, they were tested in the generator of portraits in order to assess to what extent (i) the input image remains recognisable in the typographic portrait and (ii) the input text remains legible. Figure 13 shows one of the stencils used in the creation of the portraits.

Figure 13: Stencil evolved and applied in experiment IV. The variation of the thickness of the stencil lines (left) generates a wide, continuous range of typographic weights that provide different visual emphasis (right).

A detailed description of the computational system that generates the typographic portraits is beyond the scope of this paper. Nevertheless, it is worthwhile summarising the main steps for the generation of each typographic portrait: (i) the input image is converted to grayscale; (ii) the brightness value of each pixel is calculated; (iii) an input text is composed from left to right and from top to bottom within a rectangular area proportional to the input image; (iv) for each glyph, the average brightness of the pixels located inside its bounds is calculated; and (v) the typographic weight of each glyph is set inversely proportional to the average brightness just calculated, *i.e.*, a glyph positioned over a dark area of the input image will be thicker than a glyph positioned over a lighter area.

The system that generates the portraits can be configured at different levels, *e.g.* number of text lines, leading, space between glyphs, width of the glyphs, minimum and maximum thickness of the glyphs. The adjusting of these parameters enables the generation of typographic portraits with different visual characteristics. For example, increasing the number of text lines provides more detail to the portrait; decreasing the leading and/or space between the glyphs makes the portrait visually denser; increasing the difference between the minimum and maximum thickness of the glyphs provides more contrast to the portrait. Furthermore, the system exports the generated portraits to vectors graphics. This way, one can print a postcard or a poster of his/her own typographic portrait.

Figure 14 shows typical typographic portraits created in this experiment. One can visualise animated versions of typographic portraits at http://cdv.dei.uc.pt/2018/complexity/portrait.mov. The obtained outcomes demonstrate that the stencils are able to maintain legibility and visual coherence among their glyphs while their thickness varies, which is important from a type design point of view.

The automatic evolution of new stencils enables the generation of unique typographic portraits. This reveals the potential of the evolved stencils for open-ended design projects, enabling the on-demand generation of unique typefaces.

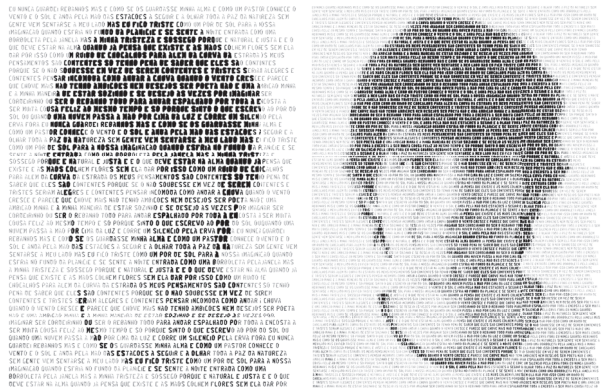

Figure 14: Typographic portraits composed of glyphs drawn with a stencil evolved with the presented framework. Two possibilities for the number of text lines are shown: 50 lines (left) and 100 lines (right). The input image is a photo of Sérgio Rebelo (graphic designer and researcher at CISUC), who tested the framework in experiment IV; and the input text is the poem "Eu Nunca Guardei Rebanhos", written in 1914 by Alberto Caeiro, an alter ego of Fernando Pessoa.

This experiment, which has been presented in a typography conference [38], demonstrates the application of the outcomes of the presented framework in a real design project, namely in the generation of typographic portraits.

4. Conclusion

An evolutionary framework for the automatic generation of type stencils was presented. We conducted a series of experiments to explore and assess this framework. Overall, the experimental results show the adequacy of the proposed framework to evolve stencils that (i) produce legible and coherent glyphs and (ii) are consistent with the preferences expressed by the user using the fitness function design interface. The results indicate that the approach guided by automatic fitness functions based on ML tend to optimise the fitness function. The results also revealed that optimising the different objectives of the fitness function will lead to legible and recognisable fonts but not necessarily aesthetically appealing. This behaviour is due to differences between how the ML techniques employed and humans perceive the inputs. Therefore, although legibility and recognisability are fundamental criteria for evolving glyphs, other visual aspects should be considered during evaluation in order to improve their aesthetic appeal.

This work demonstrated how EC and ML can inform contemporary design practices. The result is a framework that intends to provide alternative designs as stimuli for inspiration, working in a mind-opening way and promoting new ideas to create custom typefaces and letterings. This is useful, especially, when there will always be designers willing to experiment with the creation of fonts and to pursue new forms of typographic expression.

Future work will focus on: (i) implementation of adaptive mechanisms to the novelty search and archive mechanisms; (ii) creation of an archive interface to enable the user to change the behaviour of the archive mechanism during the evolutionary runs; (iii) enabling the user to save and insert evolved stencils into the evolutionary process; and (iv) implementation of the framework as web application and this way enable anyone to experiment with the evolution of type stencils.

Data Availability

The experimental results data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Funding

The first author is funded by Fundação para a Ciência e Tecnologia (FCT), Portugal [SFRH/BD/105506/2014]. We would also like to express our gratitude to NVIDIA for providing us one Titan Xp GPU to support our research.

References

[1] E. Lupton, *Thinking with Type: A Critical Guide for Designers, Writers and Students*, Princeton Architectural Press, 1st edition, 2004.

[2] I. Butterfield and M. Lewis, *Evolving Fonts*, 2000.

[3] A. Lund, "Evolving the shape of things to come: a comparison of direct manipulation and interactive evolutionary design," in *Proceedings of the International Generative Art Conference, Milan, Domus Argenia Publisher*, Rome, Italy, 2000.

[4] G. Levin, J. Feinberg, and C. Curtis, *The Alphabet Synthesis Machine*, 2001.

[5] T. Unemi and M. Soda, "An IEC-based support system for font design," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 968–973, Wash, D.C., USA, 2003.

[6] M. Schmitz, *GenoTyp, An Experiment about Genetic Typography*, 2004.

[7] M. Kuzma, "Interactive evolution of fonts," Tech. Rep., Technical University of Košice, 2008.

[8] K. Yoshida, Y. Nakagawa, and M. Köppen, "Interactive genetic algorithm for font generation system," in *Proceedings of the World Automation Congress '10*, pp. 1–6, TSI Press, 2010.

[9] T. Martins, J. Correia, E. Costa, and P. Machado, "Evotype: evolutionary type design," in *Evolutionary and Biologically Inspired Music, Sound, Art and Design*, vol. 9027 of *Lecture Notes in Computer Science*, pp. 136–147, Springer International Publishing, Cham, Switzerland, 2015.

[10] T. Martins, J. Correia, E. Costa, and P. Machado, "Evotype: from shapes to glyphs," in *Proceedings of the Genetic and Evolutionary Computation Conference '16*, pp. 261–268, ACM, New York, NY, USA, 2016.

[11] T. Martins, J. Correia, E. Costa, and P. Machado, "Evotype: towards the evolution of type stencils," in *Computational Intelligence in Music, Sound, Art and Design*, vol. 10783 of *Lecture Notes in Computer Science*, pp. 299–314, Springer International Publishing, Cham, Switzerland, 2018.

[12] P. Machado, T. Martins, H. Amaro, and P. H. Abreu, "An interface for fitness function design," in *Proceedings of the 3rd International Conference on Evolutionary and Biologically Inspired Music and Art*, J. Romero, J. McDermott, and J. Correia, Eds., Springer, Granada , Spain, 2014.

[13] P. Machado, T. Martins, H. Amaro, and P. H. Abreu, "Beyond interactive evolution: expressing intentions through fitness functions," *Leonardo*, vol. 49, no. 3, pp. 251–256, 2016.

[14] J. Mouret, "Novelty-based multiobjectivization," in *New Horizons in Evolutionary Robotics*, vol. 341 of *Studies in Computational Intelligence*, pp. 139–154, Springer Berlin Heidelberg, Berlin, Germany, 2011.

[15] A. Liapis, G. N. Yannakakis, and J. Togelius, "Sentient Sketchbook: Computer-aided game level authoring," *FDG*, pp. 213–220, 2013.

[16] A. Vinhas, F. Assunção, J. Correia, A. Ekárt, and P. Machado, "Evolutionary and biologically inspired music, sound, art and design," in *Proceedings of the 5th International Conference, EvoMUSART '16*, C. Johnson, V. Ciesielski, J. Correia, and P. Machado, Eds., Lecture Notes in Computer Science, pp. 225–240, Springer International Publishing, Cham, Switzerland, 2016.

[17] E. Kindel, "The 'Plaque Découpée Universelle': a geometric sanserif in 1870s Paris," in *Typogr. Pap. 7*, pp. 71–80, Hyphen Press, The Department of Typography & Graphic Communication, University of Reading, 2007.

[18] J. Craig, I. K. Scala, and W. Bevington, *Designing with Type: The Essential Guide to Typography*, Watson-Guptill Publications, 5th edition, 2006.

[19] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[20] P. Machado, A. Vinhas, J. Correia, and A. Ekárt, "Evolving ambiguous images," in *Proceedings of the IJCAI International Joint Conferences on Artificial Intelligence*, 2015.

[21] J. Correia, T. Martins, P. Martins, and P. Machado, "X-Faces: the exploit is out there," in *Proceedings of the Seventh International Conference on Computational Creativity (ICCC '16)*, F. Pachet, A. Cardoso, V. Corruble, and F. Ghedini, Eds., pp. 164–182, Sony CSL, Paris, France, 2016.

[22] J. Romero, P. Machado, A. Santos, and A. Cardoso, "On the development of critics in evolutionary computation artists," in *Proceedings of the Workshops on Applications of Evolutionary Computation*, Springer Verlag, Essex, UK, 2003.

[23] J. Correia, P. Machado, J. Romero, and A. Carballal, "Evolving figurative images using expression-based evolutionary art," in *Proceedings of Fourth International Conference on Computational Creativity (ICCC '13)*, pp. 24–31, Creat, 2013.

[24] P. Machado, J. Correia, and J. Romero, "Expression-based evolution of faces," in *Proceedings of the International Conference on Evolutionary and Biologically Inspired Music and Art EvoMUSART '12*, P. Machado, J. Romero, and A. Carballal, Eds., vol. 7247 of *Lecture Notes in Computer Science book series*, pp. 187–198, Springer Verlag, 2012.

[25] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR '15*, pp. 1–9, USA, 2015.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the Conference on Neural Information Processing Systems '12*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., pp. 1097–1105, Curran Associates, Inc., 2012.

[27] K. Fukushima, "Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[29] S. Baluja, D. Pomerleau, and T. Jochem, "Towards automated artificial evolution for computer-generated images," *Connection Science*, vol. 6, no. 2-3, pp. 325–354, 1994.

[30] P. Machado, J. Correia, and J. Romero, "Improving face detection," in *Proceedings of the 15th European Conference on Genetic Programming (EuroGP '12)*, A. Moraglio, S. Silva, K. Krawiec, P. Machado, and C. Cotta, Eds., pp. 73–84, Springer Berlin Heidelberg, Málaga, Spain, 2012.

[31] T. Kohonen, *Self-Organizing Maps*, Springer, Berlin, Germany, 3rd edition, 2001.

[32] T. Kohonen, "Essentials of the self-organizing map," *Neural Networks*, vol. 37, pp. 52–65, 2013.

[33] L. Wang, Y. Zhang, and J. Feng, "On the Euclidean distance of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1334–1339, 2005.

[34] A. A. Goshtasby, "Similarity and dissimilarity measures," in *Image Registration*, Advances in Pattern Recognition, pp. 7–66, Springer, London, UK, 2012.

[35] J. Lehman and K. O. Stanley, "Exploiting open-endedness to solve problems through the search for novelty," in *Proceedings of 11th International Conference on Artificial Life (ALIFE XI '08)*, MIT Press, Cambridge, Mass, USA, 2008.

[36] B. Tullett, *Typewriter Art: A Modern Anthology*, London, UK, Laurence King, 2014.

[37] F. Dietrich, "Visual intelligence: the first decade of computer art (1965-1975)," *Leonardo*, vol. 19, pp. 159–169, 1986.

[38] S. Rebelo, T. Martins, J. Bicker, and P. Machado, "Typography as image: experiments on typographic portraits," in *Proceedings of the 9th Typography Meeting*, Instituto Politécnico de Tomar, Tomar, Portugal, 2018.