Evotype: From Shapes to Glyphs

Tiago Martins
CISUC, Department of Informatics Engineering, University of Coimbra, 3030 Coimbra, Portugal
tiagofm@dei.uc.pt

João Correia
CISUC, Department of Informatics Engineering, University of Coimbra, 3030 Coimbra, Portugal
jncor@dei.uc.pt

Ernesto Costa
CISUC, Department of Informatics Engineering, University of Coimbra, 3030 Coimbra, Portugal
ernesto@dei.uc.pt

Penousal Machado
CISUC, Department of Informatics Engineering, University of Coimbra, 3030 Coimbra, Portugal
machado@dei.uc.pt

ABSTRACT

Typography plays a key communication role in the contemporary information-dense culture. Type design is a central, complex, and time consuming task. In this work we develop the generative system to type design based on an evolutionary algorithm. The key novel contributions are twofold. First, in terms of representation it relies on the use of assemblages of shapes to form glyphs. There are no limitations to the types of shapes that can be used. Second, we explore a compromise between legibility and expressiveness, testing different automatic fitness assignment schemes. The attained results show that we are able to evolve a wide variety of alternative glyphs, making the proposed system a viable alternative for real-world applications in the field of type design.

CCS Concepts

•**Computing methodologies** → **Genetic algorithms; Generative and developmental approaches;** *Object recognition; Supervised learning by classification;*

Keywords

evolutionary computation; evolutionary design; genetic algorithms; deep learning; type design

1. INTRODUCTION

The contemporary view of typography contrasts with the popular International Typographic Style that emphasises objectivity, cleanliness, readability, and the use of grotesque typefaces, such as Akzidenz-Grotesk, for almost all design projects [9]. We subscribe to the contemporary view, thus supporting that there is no universal typeface that can fit every unique design project, and that the selection of the typeface should take into account the specificity of each design project. This standpoint, along with the key communication role of typography in the contemporary information-dense culture, continuously increases the demand for innovative type design work. This, in turn, increases the need for good technological means that can assist the designer in the long laborious process of type design.

We consider that conventional computational design tools offer insufficient support to design exploration during the early conceptual stages of the type 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 work-flow 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 paper, we give a step forward of our work *Evotype* [11], an evolutionary system for type design. It employs Evolutionary Computation (EC) and Machine Learning (ML) techniques to automatically generate alternative designs for *glyphs* (see figure 1). In this context, a glyph consists in a

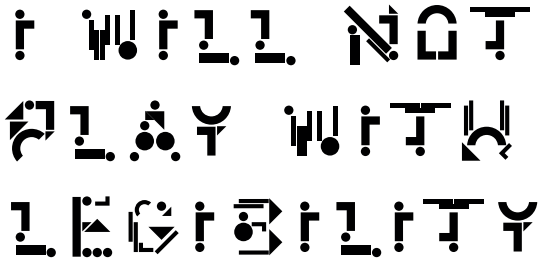

Figure 1: Text written using a font evolved by the proposed system.

GECCO '16, July 20 - 24, 2016, Denver, CO, USA

ISBN 978-1-4503-4206-3/16/07…$15.00

DOI: http://dx.doi.org/10.1145/2908812.2908907

specific design intended to represent a readable character. For instance, the lowercase letter 'e' can be embodied by different glyphs, e.g. e, e, e, e, e, e, e.

The system 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 consider that taking advantage of the creative power of EC promotes the exploration of the vast design space of innovative glyph designs, amplifying the range of possibilities and providing support to the designer in the early stages of the design process. Therefore, our goal is to develop a tool that aids the designer. The analysis of the impact of the tool, and of its outcomes, on the creative and production process of the designer is beyond the scope of this paper, but will be considered in future work.

The main contribution presented herein is a generative system capable of automatically creating glyphs using shapes. Other contributions include: (i) a generic evolutionary architecture composed of the evolutionary and evaluation modules for the generation of designs, e.g. glyphs, using shapes; (ii) the combination of Deep Learning (DL) with distance metrics to assign fitness; and (iii) an investigation into the interplay between legibility and expressiveness.

The remainder of this paper is organised as follows: section 2 summarises the related work, focusing on applications of evolutionary techniques towards the design of glyphs; section 3 thoroughly explains the behaviour of the proposed system; section 4 describes the experimental setup; section 5 provides the analysis of the experimental results; finally, section 6 presents our conclusions and directions for future work.

2. RELATED WORK

EC has been successfully applied in creative domains for the exploration of innovative solutions (e.g., [10]). Nonetheless, as far as we know, only a few evolutionary approaches for type design exist.

Butterfield and Lewis [1] employ Interactive Evolutionary Computation (IEC) to present populations of deformed letters. The letters of a given typeface are individually deformed by implicit surface primitives encoded in the genotype of each candidate individual. Lund [8] uses IEC to evolve the settings for a parametric typeface. Each parameter controls a particular visual characteristic of the typeface. Levin et al. [7] present the interactive system *Alphabet Synthesis Machine* that allows the user to generate abstract letterforms. The system employs a Genetic Algorithm (GA) to evolve a population of letterforms according to fitness metrics obtained from an initial seed glyph provided by the user. Unemi and Soda [14] propose an IEC-based system for the design of Japanese Katakana glyphs. The glyphs are constructed from simple elements that are controlled by parameters encoded in the genotype, and drawn along a pre-defined skeleton. Schmitz [12] presents the interactive program *genoTyp*, in which typefaces can be generated according to genetic rules. The program allows the user to experiment with the breeding of existing typefaces as well as the manipulation of their genes, i.e. vertexes. Kuzma [5] investigates the potential of IEC by implementing the *Font Evolving System*, which allows the user to interactively evolve fonts. Yoshida et al. [15] present the system *Personal Adapted Letter*. An Interactive Genetic Algorithm (IGA) is employed to modify the parts that define a typeface. These parts are manually defined in an earlier stage. In Martins et al. [11] we present the first iteration of *Evotype* that employs a GA to evolve glyphs for the Roman alphabet using line segments. In this early iteration, we already employ an automatic fitness assignment scheme to guide the evolutionary process.

Although some applications of evolutionary techniques for type design exist, most of them have limitations. Nearly all rely on user evaluation. The user is asked to manually guide the evolutionary process by selecting the favourite solutions in each generation until an acceptable one is obtained. Therefore, these approaches suffer from the well-known limitations of IEC, namely the user fatigue, the consequent loss of interest, and inconsistent evaluation. Additionally, some approaches require pre-existing typefaces (e.g., [1, 12, 5]) or skeletons (e.g., [14]), the drawing of initial seed glyphs (e.g., [7]), the creation of parametric typefaces (e.g., [8]), or the identification of letter parts (e.g., [15]). Finally, parametrised approaches can overly influence their outcome, since all users start from the same base, and thus are not flexible enough. The same is observed in approaches that provide a limited range of visual elements to create the glyphs (e.g., [11]).

From our analysis of the shortcomings of the related work in evolutionary type design we have extracted two main observations that guided the development of the research presented in this paper: (i) fitness assignment should be automatic, freeing the user from the need to evaluate hundreds of populations; and (ii) more than providing a final and fully functional typeface, the system should aid the designers by providing them typeface sketches that can be further refined.

3. THE APPROACH

The starting point of the proposed system is the idea of automatic assembling visual components to create glyphs for various characters. The system takes as input a Scalable Vector Graphics (SVG) file containing a set of shapes that the system uses to construct glyphs.

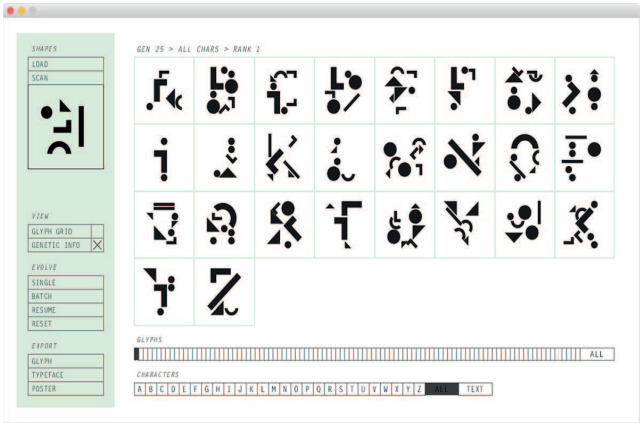

Figure 2: Snapshot of the system. A demo video can be seen at `cdv.dei.uc.pt/2016/evotype-system.mov`.

Figure 2 shows a snapshot of the system. The graphic user interface of the system is simple, as it is designed for experimental purposes. Nonetheless, it already provides the user with the necessary means to import his/her own shapes, to generate glyphs in an autonomous way, and to export them

to vector files. While the glyphs are being generated, the user can browse throughout all alternative designs and even write sentences with them, allowing him/her to test their readability.

The system integrates two modules: (i) the generation module that implements a GA to create candidate glyphs (see subsection 3.1), and (ii) the evaluation module that implements automatic fitness assignment schemes to evaluate the glyphs provided by the first module (see subsection 3.2).

3.1 Evolutionary System

The system employs a GA to generate alternative glyph designs. A GA is a stochastic search procedure inspired in the natural selection principle and in genetics [2]. In short, a GA improves a set of candidate solutions, which are initially generated at random, by iteratively employing methods of selection of the most promising for reproduction with variation. In this work, a candidate solution consists in a glyph design.

The GA is implemented to evolve glyphs for various characters. This is accomplished by evolving different populations in parallel, one for each character. Therefore, for instance, to evolve glyphs for the letters A, B, and C, the system evolves three different populations, each one being composed of glyphs that represent one character.

3.1.1 Representation

The glyphs evolved by the system are constructed from shapes provided by the user. The construction of the glyphs involve the translation, rotation, scaling, and mirroring of the shapes. The whole construction plan of each glyph is encoded in its genotype, wherein each gene encodes one shape, as well as its position, angle, scale, and if it is mirrored (see figure 3). A square grid is used to constrain the position of the shapes. Note that the order of the genes in the genotype is irrelevant at the phenotype level.

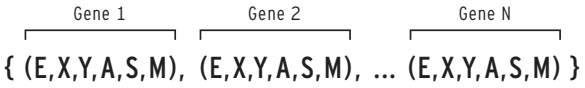

Figure 3: Genotype encoding. The genotype comprises a set of genes that consist in tuples with the attributes: shape (`E`), x-coordinate (`X`), y-coordinate (`Y`), angle (`A`), scale (`S`), and mirrored (`M`).

3.1.2 Variation Operators

New glyphs are created throughout the evolutionary process by applying variation operators such as mutation and recombination.

Mutating a candidate glyph involves stochastic modifications of some parts of its genotype. The mutation operator consists of three procedures: deletion, replacement, and insertion of genes. Each of them can occur separately with a certain probability. Therefore, the mutation operator can start by deleting a randomly chosen gene that is encoded in the genotype; then it can proceed with the replacement of genes; and lastly, it can insert a new randomly generated gene. The deletion and insertion of genes cause the variation of the number of genes encoded in the genotype, thus allowing the emergence of glyphs with different degrees of complexity.

The mutation operator is applied in a way that ensures that the genotype remains valid after mutation. A genotype is considered valid if (i) the number of genes and the values of the encoded attributes remain within preset ranges; (ii) all its genes are different; and (iii) the shapes represented by its genes are located inside the bounds of the grid.

For recombination, an uniform crossover operator is implemented. Two offspring are generated by sequentially copying genes with the same probability from one of the two parents. Since the genotypes of the parents can have different lengths, the crossover operator shuffles the genes inside each genotype before recombining them.

3.2 Evaluation

We adopt an automatic fitness assignment scheme to autonomously guide the evolutionary process. In this work we test three different fitness functions, which are detailed in this subsection.

3.2.1 Root Mean Square Error

The first fitness function evaluates a candidate glyph based on its visual similarity to an existing glyph. The similarity between two glyphs is calculated using a Root Mean Square Error (RMSE) method that measures how close the candidate glyph is to the reference glyph on a pixel-by-pixel basis. Thus, for a candidate image C and a target image T we have,

$$\mathrm{RMSE}(C,T) = \sqrt{\frac{\sum_{i=1}^{S_C}(c_i - t_i)^2}{S_C}} \tag{1}$$

where S_C is the size of image C.

To guide evolution we create a fitness function that is suitable for maximisation. Furthermore, we apply a logarithmic scaling operation (see subsection 3.2.3), which results in the fitness function depicted in equation 2.

$$\mathrm{fit}_{\mathrm{RMSE}}(C,T) = \log_2\left(1 + \frac{1}{1+\mathrm{RMSE}(C,T)}\right) \tag{2}$$

3.2.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a type of Deep Neural Networks (DNNs) that have been used successfully in image classification and recognition tasks [6]. The main characteristic of a CNN is the usage of convolutional and pooling layers, which provides feature extraction and dimensionality reduction in training [3]. Each layer can be seen as a filter from which features are extracted and learnt. In recent years, it has consistently attained good results in several supervised image related classification tasks [4, 13].

Since we are interested in evolving a legible composition of shapes, the recognisability of each glyph becomes an important factor. As such, we develop and train CNN classifiers for character recognition. The output of each classifier indicates its confidence in recognising a given input image as its desired character. An output of 1 indicates total confidence while an output of 0 indicates the opposite. As such, to guide evolution we employ the following fitness function, which also includes a logarithmic scaling operation:

$$\mathrm{fit}_{\mathrm{CNN}}(C) = \log_2(1 + \mathrm{CNN}(C)) \tag{3}$$

3.2.3 Hybrid

The third fitness function consists in a weighted sum of the values returned by the two previous fitness functions. The theoretical optimum is a glyph that simultaneously maximises fit_{RMSE} and fit_{CNN} based fitness. However, this can be impossible and, in those circumstances, we want evolution to find a compromise between these two components, avoiding solutions that focus only on one of these fitness measures. The inclusion of the logarithmic scaling operation is intended to address this issue, promoting the improvement of fitness through the exploration of both criteria. As one of the components increases, e.g. the fit_{CNN}, the gains in fitness decreases, meaning that it becomes more advantageous to focus on the improvement of the component that is farther from the maximum value, promoting the discovery of compromise solutions.

The hybrid fitness function is as follows:

$$\text{fit}_{\text{Hybrid}} = w_0 \times \text{fit}_{\text{RMSE}}(C, T) + w_1 \times \text{fit}_{\text{CNN}}(C), \qquad (4)$$

where w_0 and w_1 specify the weights given to each component.

4. EXPERIMENTAL SETUP

We perform experiments on the proposed system with three major goals in mind. First, we assess if, and to what extent, the system can generate glyphs for different characters using a given set of shapes. Second, we investigate the impact of the use of different fitness functions (see subsection 3.2) on the quality of the resulting glyphs. Finally, we analyse the variety of the glyphs that can be generated.

In the experiments presented in this work, we evolve glyphs for all the uppercase letters of the Latin alphabet. We experiment the generation of glyphs using several sets of shapes, however, due to space limitations, the results presented herein are produced using the shapes shown in figure 4. The experimental parameters used in these experiments are summarised in table 1.

Figure 4: Experimental shapes.

As target font for fit_{RMSE}, we choose one of the most used fonts of the Google Fonts platform, that is *Roboto*. More specifically, we use the medium style of this font.

Concerning the classifier, we train a CNN based on the topological model of the Alexnet model of [4] as an off-the-shelf topology. The Alexnet model is a CNN with five convolutional layers, with max-pooling layers, and three fully-connected layers with a final 1000-way softmax, known for attaining the best result on ILSVR 2012 competition in image recognition, and for its contribution to the development and training of CNNs.

Based on the preliminary experiments [11], we isolate each character and train against random images. The underlying idea is for the CNN to learn the features of each character against other types of images and not exclusively against the other characters in order to create a general purpose character classifier. Thus, we train 26 classifiers with two classes: positive and negative. For the positive class dataset, we gather a total of 537 Latin fonts served by Google Fonts[1].

[1] `github.com/google/fonts`

Evolutionary System		CNN Training	
Runs	30	Solver	Stochastic Gradient Descent (SGD)
Generations	200		
Population size	100	Epochs	1000
Elite size	1	Learning rate	0.01
Selection method	Tournament	Learning rate policy	Drop 0.1 every 500 epochs
Tournament size	2		
Recombination rate	0.7	Momentum	0.9
Mutation del. rate	0.05	Positive samples	200
Mutation replac. rate	0.05	Negative samples	200
Mutation insert. rate	0.05		
Glyph grid size	13 × 13		
Max. genes	8	**Evaluation**	
Allow overlaps	No	RMSE target font	Roboto Medium
Rotation angles	8	Hybrid w_0	0.5
Scales	1; 2	Hybrid w_1	0.5

Table 1: Experimental parameters.

The negative class dataset is built with Creative Common images from Flickr by searching with the following keywords: random, images, and photos. The images of both datasets, are resized to 32 by 32 pixels and converted to greyscale. The remaining relevant training parameters are presented in table 1. For each glyph, 200 instances of the positive and negative sets are randomly selected for training purposes. After training, we test the networks against the remaining instances of the positive and negative dataset: 347 and 3871, respectively. The average accuracy attained for the training dataset is 0.998 and for the test dataset is 0.9975.

5. EXPERIMENTAL RESULTS

In this section we present and analyse the experimental results. First, we focus on the analysis of the evolution of fitness when using different fitness functions. Then, we examine how the use of different fitness functions affects genotype length and shapes diversity throughout the evolutionary process. Finally, we consider the visual appearance of the evolved glyphs.

Figures 5, 6, and 7 depict the evolution of fitness of the best individuals throughout the generations of the runs conducted using, respectively, fit_{RMSE}, fit_{CNN} and $\text{fit}_{\text{Hybrid}}$ as fitness functions. The values that these individuals would obtain with the remaining fitness functions are also calculated and plotted, which allows us to examine, for instance, how evolving with the goal of maximising fit_{RMSE} affects fit_{CNN} and $\text{fit}_{\text{Hybrid}}$.

As it can be observed from the perusal of these charts, the EC algorithm is able to optimise the fitness function, guiding evolution. They also reveal that maximising fit_{CNN} is significantly easier than maximising the other fitness measures. Analysing the evolution of fit_{CNN} and $\text{fit}_{\text{Hybrid}}$ in the runs guided by fit_{RMSE} allows us to state that using fit_{RMSE} is sufficient to attain, in the long run, high values for all fitness functions. The same cannot be stated for the runs where fit_{CNN} is used to guide evolution. In these runs the EC engine quickly finds glyphs that the CNN classifies, with certainty, as belonging to the desired class, which causes early convergence. Although fit_{RMSE} and $\text{fit}_{\text{Hybrid}}$ attain relatively high scores, they never reach their maximum values. Finally, as expected, when $\text{fit}_{\text{Hybrid}}$ is used to guide evolution, high values for all fitness functions are attained

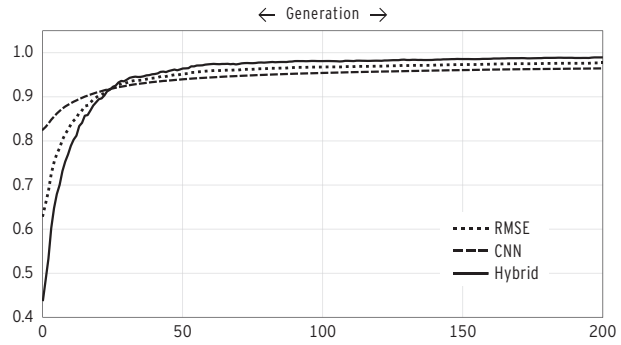

Figure 5: Evolution of the fitness of the best individual across generation, when using fit_{RMSE} to guide evolution. The results are averages of 30 runs.

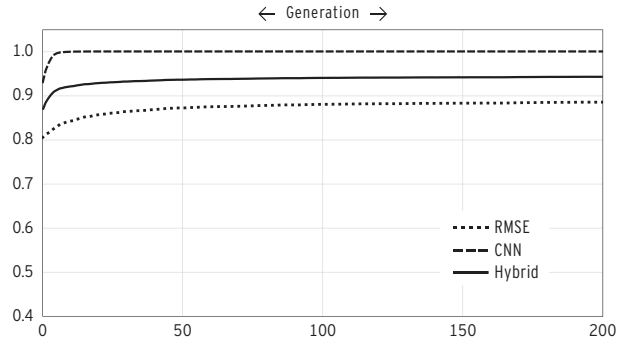

Figure 6: Evolution of the fitness of the best individual across generation, when using fit_{CNN} to guide evolution. The results are averages of 30 runs.

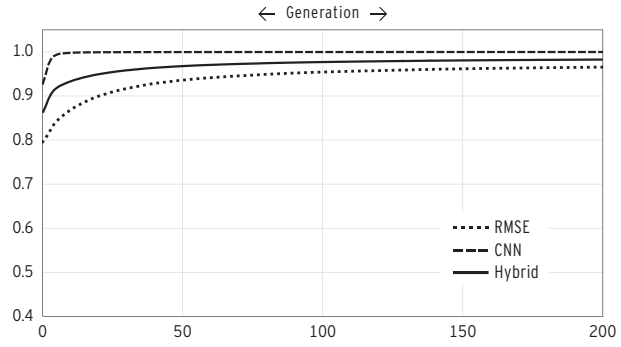

Figure 7: Evolution of the fitness of the best individual across generation, when using $\text{fit}_{\text{Hybrid}}$ to guide evolution. The results are averages of 30 runs.

in few generations, and, in the long run, values close to their respective maximums are obtained.

In what concerns the fitness of the best individuals of the last generation, the differences among the runs conducted with fit_{RMSE} and $\text{fit}_{\text{Hybrid}}$ are almost not noticeable for any of the fitness measures that we consider. However, differences can be perceived in terms of the dynamics of the runs, particularly in the initial generations. There are also differences both in terms of dynamics and final fitness values between the runs guided by fit_{CNN} and the remaining ones.

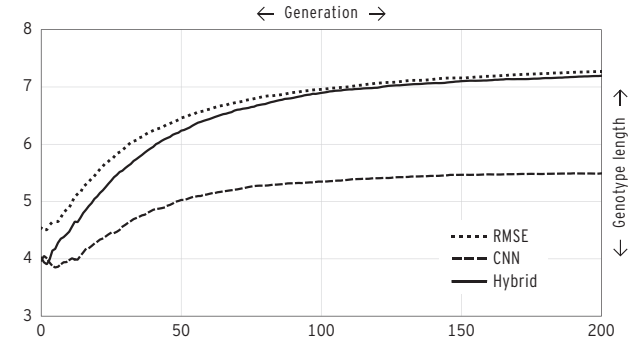

Figure 8: Progression of the genotypes' length of the fittest glyphs over the generations. The visualised results are averages of 30 runs.

Figure 8 depicts the evolution of the length of the genotypes of the fittest glyphs. As it can be observed, due to the need of matching a given target as precisely as possible, the runs guided by fit_{RMSE} and $\text{fit}_{\text{Hybrid}}$ tend to use a high number of shapes. In contrast, the glyphs evolved in the runs guided by fit_{CNN} only need to be recognised as the desired glyph, meaning that a fewer number of shapes is needed.

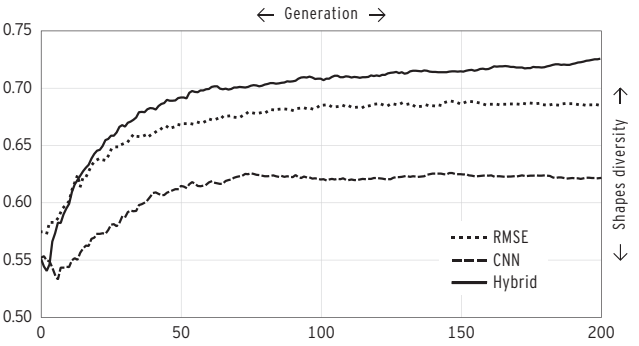

Figure 9: Progression of the shapes' diversity of the fittest glyphs over the generations. The visualised results are averages of 30 runs.

Finally, we focus on the diversity of the set of shapes used, calculating the percentage of the shapes of the set provided by the user that is used in the fittest individual of each generation. As figure 9 shows, the runs based on fit_{CNN} result in glyphs using a less diversified set of shapes. The runs using fit_{RMSE} result in glyphs with an intermediate degree of shapes diversity, while the ones evolved using $\text{fit}_{\text{Hybrid}}$ are the ones that promote the use of a more diversified set of shapes. An explanation for the observed differences follows. When using fit_{CNN}, the EC engine is only required to evolve

glyphs recognised as the target character, there is no reason to believe that such glyphs will be composed of a varied set of shapes, since the repetition and transformation in terms of position, scaling and rotation of a single shape is sufficient to meet this requirement. When fit_{RMSE} is employed, it becomes necessary to use a more diversified set of shapes to match, as precisely as possible, the target image, since different areas of the image may require the use of different shapes.

Therefore, the real question is why using a combination of these measures to guide evolution seems to result in a higher diversity. The explanation appears to be related with the dynamics of the evolutionary process. As we have previously mentioned, it is easier to improve fit_{CNN} than fit_{RMSE}, as such in the early stages of the evolutionary process the fittest glyphs tend to be the ones focusing on fit_{CNN}. This may lead to the inclusion of shapes in the genotype that are not ideal for the purpose of maximising fit_{RMSE}. These shapes would not be included if fit_{RMSE} was used to guide evolution. In a later stage, since $\text{fit}_{\text{Hybrid}}$ also takes fit_{RMSE} into account the shapes that fit_{RMSE} tends to use will gradually make its way into the genotype, gradually increasing the diversity of the shapes used in the construction of the glyphs. Informally, when using $\text{fit}_{\text{Hybrid}}$ the fit_{CNN} tends to provide the skeleton for the glyph, while fit_{RMSE} provides additional visual detail.

Next, we will focus on the analysis of the visual appearance of the evolved glyphs. Figure 10 shows typical glyphs generated using each fitness function. It is noticeable the differences at the visual level. For instance, the evolved glyphs with fit_{RMSE} function reinforce the idea that the evolutionary process is trying to use and place as many shapes as necessary to cover the target image, and thus maximise its fitness. In contrast, the fit_{CNN} based glyphs are more abstract. They have a reduced number of shapes, which causes the glyphs to have some missing parts. This fact is consistent with the results attained in terms of fitness function, genotype length and shapes diversity. The $\text{fit}_{\text{Hybrid}}$ showcases a compromise between the abstraction of the fit_{CNN} and the complete glyph structure promoted by the fit_{RMSE}.

The plasticity of the glyphs is illustrated in figure 11. As it can be perceived, all the glyphs are different. In the fit_{RMSE}, although a target image is used, the different shapes that are used to construct the glyphs provide them visual diversity. In the same figure, we can see how each fitness function influences the diversity of the shapes that compose the glyphs. The results obtained using the fit_{CNN} do not differ much from the ones generated using fit_{RMSE} or $\text{fit}_{\text{Hybrid}}$, and tend to repeat the same shapes as depicted in figures 8 and 9.

Figure 12 shows the evolution of the uppercase letter E throughout the generations for each fitness function. It is visible the different evolutionary process promoted by the fit_{RMSE} and the fit_{CNN}, i.e. the first one promotes the use of small shapes to fill the content of the target image, while the second one promotes the accurate placement of shapes and the use of less shapes. The $\text{fit}_{\text{Hybrid}}$ stands out by attaining legible glyphs at earlier generations using a more diverse set of shapes.

The different alphabets presented in figure 13 highlight the diversity of glyphs that can be evolved by the system. By allowing the user to input his/her own shapes to build the font, the system is able to provide identity to the evolved glyphs and therefore keep the individual touch of the designer. Furthermore, although the glyphs are evolved in

Figure 10: Typical glyphs evolved using the three different fitness functions.

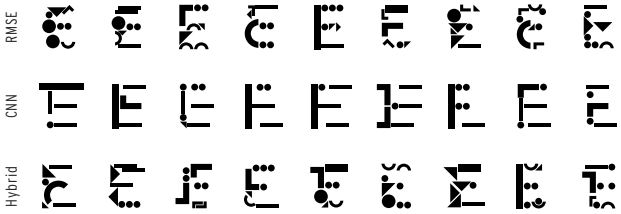

Figure 11: Typical glyphs for the letter E at the generation 200 using the three different fitness functions.

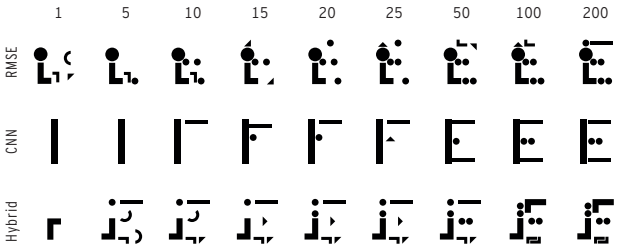

Figure 12: Typical evolution of the fittest glyphs for the letter E using the three fitness functions.

Figure 13: Examples of alphabets evolved in different runs. More evolutionary runs can be visualised at `cdv.dei.uc.pt/2016/evotype-runs.mov`.

separated populations, we consider that the use of the same set of shapes to construct all glyphs allows the emergence of a common visual style.

The system is able to generate unusual glyphs that push the boundaries between expressiveness and legibility, and encourage graphic experimentation in type design. The generated glyphs may be suitable for projects that require fonts that have the ability to communicate an identity or concept, thus mediating something else than language. The system has also potential for open-ended design projects, allowing for instance the on-demand generation of unique fonts.

The user can export the evolved glyphs and use them to create typographic compositions in most common design software tools. Additionally, the modular architecture of the system (evolutionary and evaluation modules) enables its extension and adaptation to other graphic design problems.

6. CONCLUSIONS AND FUTURE WORK

We have described and tested an iteration of *Evotype*, a generative system for the automatic creation of glyphs. The experimental results demonstrate the ability of the system to generate a wide variety of alternative glyphs that push the boundaries between legibility and expressiveness. The results also show the impact of different fitness functions on the resulting glyphs.

There are future enhancements from which the proposed system can benefit, including: (i) experimenting different weights in the hybrid fitness function; (ii) considering the visual coherence, i.e. the common visual characteristics, between glyphs by exploring other approaches, e.g. Genetic Programming; (iii) supporting a more active participation of the user during the evolutionary process, e.g. allowing the user to modify evolved glyphs and insert them back into the population; and (iv) conducting user studies to validate the utility of the tool, to evaluate the effectiveness of the tool with which users can generate glyph designs, and to verify their aesthetic and functional quality.

7. ACKNOWLEDGEMENTS

This research is partially funded by: Fundação para a Ciência e Tecnologia (FCT), Portugal, under the grants SFRH/BD/90968/2012 and SFRH/BD/105506/2014; and project ConCreTe. The project ConCreTe acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET grant number 611733.

8. REFERENCES

[1] I. Butterfield and M. Lewis. Evolving fonts, 2000. Consulted in http://accad.osu.edu/~mlewis/AED/Fonts/ on December 2015.

[2] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Natural computing series. Springer, Berlin, Heidelberg, Paris, 2015.

[3] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[5] M. Kuzma. Interactive evolution of fonts. Master's thesis, Technical University of Košice, 2008.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[7] G. Levin, J. Feinberg, and C. Curtis. The alphabet synthesis machine, 2001. Consulted in http://www.alphabetsynthesis.com on December 2015.

[8] A. Lund. Evolving the shape of things to come: A comparison of direct manipulation and interactive evolutionary design. In *International Conference on Generative Art*. Domus Argenia, Rome, Italy, 2000.

[9] E. Lupton and J. C. Phillips. *Graphic design: the new basics*. Princeton Architectural Press, 2008.

[10] P. Machado, J. Romero, and B. Manaris. Experiments in computational aesthetics: An iterative approach to stylistic change in evolutionary art. In J. Romero and P. Machado, editors, *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*, pages 381–415. Springer Berlin Heidelberg, 2007.

[11] T. Martins, J. Correia, E. Costa, and P. Machado. Evotype: Evolutionary type design. In C. Johnson, A. Carballal, and J. Correia, editors, *Evolutionary and Biologically Inspired Music, Sound, Art and Design*, volume 9027 of *Lecture Notes in Computer Science*, pages 136–147. Springer International Publishing, 2015.

[12] M. Schmitz. genotyp, an experiment about genetic typography. Presented at Generative Art Conference 2004, 2004.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[14] T. Unemi and M. Soda. An iec-based support system for font design. In *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics: Washington, D.C., USA, 5–8 October 2003*, pages 968–973, 2003.

[15] K. Yoshida, Y. Nakagawa, and M. Köppen. Interactive genetic algorithm for font generation system. In *World Automation Congress, 2010*, pages 1–6. TSI Press., 2010.