Novelty and Figurative Expression-Based Evolutionary Art

· U C ·

Adriano Rua Vinhas

avinhas@student.dei.uc.pt

Faculty of Sciences and Technology

University of Coimbra

Advisors

Fernando Jorge Penousal Martins Machado

Anikó Ekárt

Coimbra 2015

Examiners

Prof. Ernesto Jorge Fernandes Costa
Prof. Fernando Boavida

Acknowledgements

This research is partially funded by 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.

I would like to acknowledge all my gratitude to my advisors, Penousal Machado and Anikó Ekart, for all their guidance and support in this journey, and specially for their endless effort in providing all the resources necessary to fulfil my personal goal of having a different work experience abroad.

I want to thank also to the CDV group, for their cooperation, feedback and support throughout the year.

A special thanks to my friends Adriana Fernandes, António Marques, Sara Cardoso and Teresa Alves. Each one at its turn had the patience to encourage me, awake my desire to work abroad, listen to my complains and help me with their opinions and feedback.

Last, but most important of all, I would like to thank my family, specially my parents. Since the diapers time they have been dealing with a grumpy human being like me. Nevertheless, they educated me with a lot of effort, dedication and without requiring anything in change. This work is also their work and they truly deserve every word I wrote.

Adriano Vinhas
Coimbra, July 2015

Abstract

Evolutionary Art combines evolutionary computation approaches and computer graphics in order to generate artworks. Within this field, it has been shown that it is possible to evolve interesting artworks that can please the human eye. However, most of these artworks are created with human supervision and the large majority of artworks generated are abstract.

In contrast, this dissertation is inspired on a previous work that aims to evolve figurative images, that is, images that resemble some object to the human eye, without any human supervision during the process. Instead, the supervision process resorts to an object classifier, trained to recognize specific objects, which is used to assign fitness.

In this dissertation, this work is expanded in order to demonstrate that, depending on the object classifiers created, it is possible to generate any object. Furthermore, this dissertation explores the evolution of ambiguous images, that is, images that can resemble several objects at the same time within the same region, using a set of classifiers to evaluate instead of just one. Finally, this dissertation also focuses on the evolution of images that can be considered as different as possible among them, using novelty search techniques.

Experimental results show that it is possible to generate images that are (1) figurative and (2) ambiguous, from a computational and human perspective. The results also indicate that the classifiers' robustness plays an important role in approximating the computational and the human point of view. Furthermore, the success in employing novelty mechanisms depends also on the classifiers used: when a single classifier is used to guide evolution, novelty search tends to result in a broader set of diverse images. However, when several classifiers are used, novelty search is only able to promote diversity when the classifiers are permissive.

Resumo

Arte Evolucionária é um campo de estudo que combina técnicas de computação evolucionária e computação gráfica, de modo a gerar arte visual. Dentro deste campo, já foi demonstrado que é possível evoluir imagens interessantes que agradam ao utilizador. No entanto, a maioria destes trabalhos são criados com a supervisão de um utilizador e são imagens maioritariamente abstractas.

Por outro lado, esta dissertação é inspirada em trabalhos anteriores que evoluem imagens figurativas, que são imagens que pretendem representar um dado objecto, sem a supervisão humana. Em vez disso, o processo de supervisão é controlado com a ajuda que de um classificador, treinado para reconhecer um dado tipo de objectos, que é usado para medir a qualidade das imagens.

Nesta dissertação, este trabalho é estendido de modo a demonstrar que, dependendo dos classificadores usados, é possível gerar qualquer objecto. Para além disso, esta dissertação explora a evolução de imagens ambíguas, isto é, imagens que são capazes de representar vários objectos simultameamente dentro da mesma região, usando um conjunto de classificadores em vez de um só. Finalmente, esta dissertação também se debruça na evolução de imagens que possam ser consideradas o mais distintas possível entre elas, recorrendo a técnicas de pesquisa de novidade.

Os resultados experimentais mostram que é possível gerar imagens que (1) são figurativas e (2) são ambíguas, de um ponto de vista computacional e humano. Os resultados demonstram que a robustez dos classificadores é importante para aproximar os pontos de vista computacional e humano. Para além disso, o sucesso na utilização de mecanismos de novidade depende dos classificadores usados: quando um só classificador é usado para guiar a evolução, a pesquisa de novidade tende a obter um conjunto mais alargado de imagens. No entanto, quando são utilizados vários classificadores, a pesquisa por novidade apenas é capaz de promover diversidade quando são utilizados classificadores mais permissivos.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Based in Genetic Programming [1], Karl Sims got inspiration to create images from programs evolved with the help of the user [2], creating an approach within the field of Evolutionary Art, known as expression-based.

Since that moment, many issues and challenges were addressed regarding this topic. Sims' work relies on the user to select the best images in each generation, which causes several limitations in the final artworks. Due to these limitations, a general purpose challenge arose in this field: automate the evaluation process, whether one wants to evolve images according to an aesthetic criteria or towards a goal image. Many works developed regarding Evolutionary Art were inspired by Sims' work, causing them to share the same shortcomings pointed out in Sims' work.

Additionally, in McCormack's work [3], a limitation of expression-based imagery was mentioned, namely expression-based images, would tend to hold the same abstract aspect, and they would tend to share the same "class", clearly showing that they were computer-generated, by mathematical functions. Although this is true from a practical point of view, Machado and Cardoso proved that it was theoretically possible to generate any image using an expression-based approach [4].

Among the literature regarding Evolutionary Art, the majority of the works relies on the user to generate new artworks. Within the works that tried to produce imagery autonomously, most of them focus on the creation of abstract images evaluated according to a given hardcoded criteria. Moreover, the few works which tackled figurative artworks, evolved images towards a "goal image" or towards a restricted set of objects. On the other hand, this work intends to address one of the biggest challenges in Evolutionary Art, mentioned before, which is to automate the evaluation process of images, creating non-abstract artworks, in opposition to most of the works, which focus in abstract imagery. Additionally, this work also innovates by addressing the problems of evolving images that represent several different objects and evolving ambiguous images for the first time.

The work of Correia and Machado produced results which are in line with this work's intentions by using an expression-based approach to generate autonomously figurative images [5], which not only hits the challenge of relieving the user from evaluating lots of images, but also reveals a distinctive value by evolving non-abstract images through the resemblance of a given image to an object.

Although they proved that their approach could be extended to other objects rather than faces, by achieving promising results, they identified a major flaw in their results: the lack of variety in terms of objects generated within the same seed. Furthermore, more study needs to be performed to determine how far can this approach be explored in order to achieve more interesting results and more diverse artworks. Consequently, this work applies and refines their proposed approach, inspired by novelty search ideas, in an attempt to overcome the flaws that they identified.

The assessment of this approach involves the creation of an original and wide set of object classifiers using hand built datasets, the creation of a framework which is able to reproduce their approach, and an analysis of the results, in terms of ability to evolve effectively the desired objects, variety of artworks produced and subjective judgements regarding the resemblance of the final artworks to the expected objects.

1.1 Scope

The work of Correia and Machado [5] has a distinctive value when compared to others within the field of Evolutionary Art, because they use an expression-based approach to create figurative images autonomously. In order to assess automatically the quality of those images and their resemblance to a given "target class", an object classifier is incorporated within their approach. This work is built upon Correia and Machado's findings, because there is a desire to evolve the same type of artworks using a similar process.

In their work, the authors say that what makes the approach viable and easy to apply is the existence of off-the-shelf classifiers[1]. However, being restricted to third party sources is not desirable when one wants to prove the generalization of their approach. Consequently, its real power can be put to test through the creation of new classifiers and a trial to create different and diverse artworks.

[1]Object detection classifiers which were trained, validated and provided by external sources

The new classifiers created within the scope of this work, despite they aim to represent several objects, they share the same classifying technique, used in previous works of Correia and Machado and originally formulated by Viola and Jones [6]. Applying this technique, new classifiers are created by choosing a set of parameters and building customized datasets. So, this work does not intend to study different classifying schemes.

Regarding the artworks produced, this work is restricted to expression based evolutionary art and no aesthetic nor complexity judgements are made over the images. The only evaluations performed are in terms of resemblance to a given object and in terms of uniqueness.

1.2 Objectives

Following in the footsteps of previous studies which aim to explore automatic creation of artworks, this work involves expression based evolutionary art and it can be summed up in three main goals:

- Autonomous creation of figurative images, using evolutionary algorithms to evolve them and an object classifier to help in determining their quality;
- Autonomous creation of ambiguous images, using evolutionary algorithms to evolve them and multiple object classifiers to determine their ambiguity level;
- Study and development of novelty search mechanisms to apply in the creation process of artworks, by including a dissimilarity metric to influence their evolution.

As an ultimate goal, this work is expected to produce scientific results worthy of dissemination in specialized conferences.

It is also expected that the implementation work on image rendering and automatic fitness assignment can be disseminated and become widespread over the community.

1.3 Outline

The remainder of this document is organized as follows: Chapter 2 contains the state of the art, describing Evolutionary Computation concepts, with a special focus on Genetic Programming and a short overview on Multi Objective Evolutionary

Algorithms, followed by diversity metrics, and finally Evolutionary Art. Chapter 3 displays the work performed during the first semester, firstly describing the approach used to obtain expression-based figurative images and then explaining step by step the actions taken in order to generate images, presenting examples of obtained results. Then, Chapter 4 addresses the evolution of ambiguous images, explaining which modifications were made to the original approach, describing how the fitness functions from different classifiers were aggregated into a combined fitness function, and presenting the experimentation work performed. Chapter 5 focuses on the novelty study. It starts by explaining again new modifications done in the original approach, concepts about the novelty operation mode within the approach and their respective way of computation, and presents the results obtained regarding the novelty study. Finally, Chapter 6 contains the final considerations about the work performed until now and also points new directions of this study.

Chapter 2

State of the Art

This chapter aims to present important literature related to the planned work and research objectives.

Initially, section 2.1 presents a brief introduction to Evolutionary Computation (EC), describing in section 2.1.1 the inspiration, principal components and mechanisms involved with this field of study, including a special focus in Genetic Programming in section 2.1.2 (the EC method that is going to be used) and an overview in multiobjective evolutionary algorithms in section 2.1.3. Then, section 2.2 addresses the diversity mechanisms considered for further exploration in this work. It focuses on Novelty Search in section 2.2.1, Fitness Sharing in section 2.2.2 and it describes distance metrics used along with these techniques while studying related work in the context of Evolutionary Art and image generation (in section 2.2.3).

Finally, section 2.3 presents a background in the Evolutionary Art field. This section is divided in three subsections. Section 2.3.1 covers interactive approaches in the context of Evolutionary Art, section 2.3.2 covers approaches which are able to evolve images automatically without any supervision and section 2.3.3 is more focused on the specific branch of Evolutionary Art imagery, the evolution of expression-based figurative images.

2.1 Introduction to EC

In 1859, Charles Darwin published the first edition of his book “On the Origin of Species”, where he introduced a new scientific theory which suggested an explanation for the biological diversity we can see and its mechanisms, based on the evidence that Darwin gathered during the Beagle expedition [7].

In essence, Darwin’s theory explained the process of natural evolution of species around the concept of **natural selection**. Basically, according to the principles of natural selection, nature itself and its resources have an essential role to select the fittest individuals, the ones who have more capability to achieve those resources and adapt to the current external conditions (the **environment**). These individuals are the ones with major probability of survival and therefore reproduce between themselves to pass their characteristics to their offspring. On the other hand, individuals with lower degree of adaptation to the environment die without leaving their characteristics to future generations.

To be more precise, the characteristics that determine whether an individual is more or less adapted to the environment are its behavioural or physical features. These types of features, visible to the human eye, are defined as the **phenotype** of an individual.

However, from a microscopic point of view, the phenotype of an individual is a result of the encoding of the **genotype**, that is, the genetic material, organized atomically in **genes**, containing the information responsible for the characteristics expressed by an individual. The range of values that a gene can assume is defined as the **alleles**.

As explained before, the fittest individuals tend to pass their characteristics through a breeding process which allows further generations to become better adapted to the environment. This process occurs at the microscopic level, consisting simply in creating a new individual using genetic material from the parent(s), which implies that variations that might occur in the next generation act on the individual's genotype.

This theory was further explored in different ways, creating new branches and fields of study. Evolutionary Computation (EC) [8, 9, 10, 11] started as a field in computer science, more specifically computational intelligence, taking its biological inspiration seeded by Darwin and his Natural Selection Theory.

The connection between Darwin's ideas and Artificial Intelligence was explored by several authors during the 20th century. John Holland's work will be discussed in detail as he pioneered the evolutionary approach known as genetic algorithms and formulated the **schema theory** [12]. He centered his work around the concept of adaptation. However, he enunciated a different definition, which was not restricted to the biology field only:

"Adaptive processes have a critical role in fields as diverse as psychology ("learning") economics ("optimal planning"), control, artificial intelligence, computational mathematics and sampling ("statistical inference"). Basically, adaptive processes are optimization processes (...)"[12]

Based on this definition, Holland formulated a general framework capable of optimizing processes, unifying different fields, as he did in his definition of adaptive processes. In simple words, this framework consisted in an environment E (nature), a set of structures α (the individuals) which were measured according to a performance measure μ (their chance of survival). α and μ were submitted to an adaptive plan τ (the simulation of natural selection and breeding operations) that determined modifications to be performed over the structures in response to the environment. This process was repeated using discrete instants of time t (generations).

Holland also contributed one of the most important advantages of using EC: one can apply an EC approach to a wide range of areas since it is domain-independent. An EC algorithm can be applied to a problem where the optimal solution is hard to find in a large and undefined search space, due to time or computational resources limitations. Instead of waiting for the global optimal solution to be evolved, the algorithm can be stopped when an suboptimal (but acceptable) solution is found.

Nowadays, EC is studied in the field of Computer Science and there are two main techniques: Genetic Algorithms (GA) [12, 13, 14] and Genetic Programming (GP)[1, 15, 16, 17], being briefly described as a stochastic process of solving problems using biology as an analogy.

In the remainder of this section, 2.1.1 will describe the main components and procedures needed to create a generic evolutionary algorithm whilst section 2.1.2 will focus on Genetic Programming, since it was the EC approach chosen for the work. Finally, evolutionary algorithms with several objectives are addressed in section 2.1.3, where some techniques of selecting the best individuals according to multiple criteria are tackled.

2.1.1 The Generic Evolutionary Algorithm

EC has a wide range of approaches which can be used to solve a problem (some described in 2.1). However, an Evolutionary Algorithm has a common set of procedures and components shared between all approaches.

First of all, they work with a set of individuals (a **population**), where each individual is located in a give point of the search space as a candidate solution to solve a problem. This population is usually created randomly and is submitted to the environment pressure (survival of the fittest) over a number of generations. During the evolution process, in each generation all the individuals are measured in terms of capability to solve the problem using a quality measure called evaluation/fitness function.

This evaluation allows the application of a selection operator which chooses stochastically the best individuals, the ones that will be allowed to breed (**parents**) and create offspring. Then, the variation operators (typically crossover and/or mutation) are applied, in order to obtain increasingly more promising solutions while hoping to maintain the necessary diversity to explore the search space globally. Crossover (or recombination) consists in choosing two individuals from the parents group and then generating one or more descendants by exchanging genetic material between the parents. Mutation can be posteriorly applied over the individuals, by modifying, with some probability, genes contained in an individual's genotype, creating the children set (**offspring**). After applying the variation operators, the survivor selection mechanism chooses which individuals will persist for the next generation, filling the new population with individuals from the parents or the offspring.

The Evolutionary Algorithm runs iteratively, generation by generation, until some termination criterion causes the algorithm to end. The diagram in figure 2.1 describes the typical flow of an Evolutionary Algorithm. Algorithm 1 explains the operation of an Evolutionary Algorithm through pseudocode.

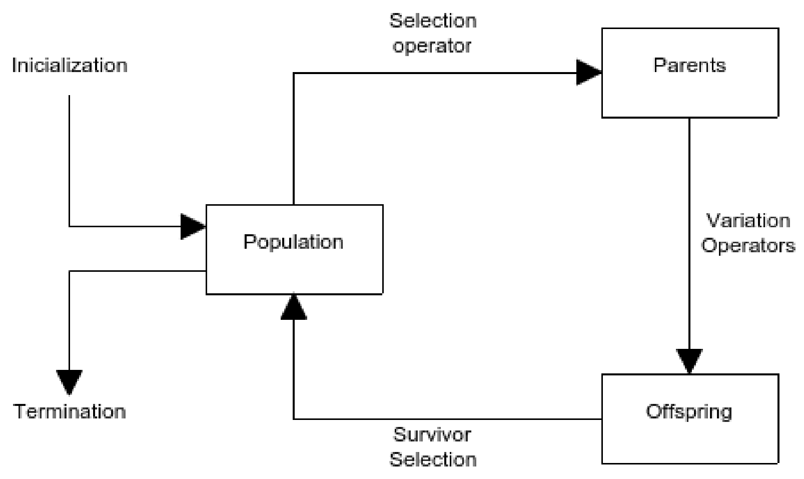


Figure 2.1: Flow of a generic Evolutionary Algorithm [8].

Evolutionary algorithms have the below specific characteristics:

- They work at the population level, with several solutions simultaneously;

```
Procedure: GenericEvolutionaryAlgorithm
begin
    population ← initializePopulation();
    population ← evaluateIndividuals(population);
    while terminationcriteria not satisfied do
        parents ← applySelectionOperator(population);
        offspring ← recombinationOperator(parents);
        offspring ← mutationOperator(offspring);
        offspring ← evaluateIndividuals(offspring);
        population ← survivalsSelector(offspring, population);
    end
end
```

Algorithm 1: Generic Evolutionary Algorithm.

- They belong to the "generate-and-test" algorithms, in which generate corresponds to the initialisation and creation of new and hopefully better individuals than the ones created before ,and test corresponds to the evaluation of the adaptability to the environment;
- They are stochastic algorithms, and for this reason they do not guarantee an optimum solution.

In order to contribute to the success of an Evolutionary Algorithm, several factors must be taken into account. Those factors will be enumerated and further described:

- Representation
- Fitness Function
- Population
- Selection Operators
- Variation Operators
- Survivor selection mechanism
- Initialization method
- Termination criteria

Representation

Representation means how the individual solutions are encoded computationally, so it is the bridge between the genotype space and the phenotype space. Similarly to nature, the genotype contains the genetic material of an individual, which in EC is the computer code that can take the form of a bit-string, tree, graph, etc. depending in the form of EC applied. In EC, this defines the search space where the evolution takes place, that is, where individuals breed and exchange genetic material. This artificial genetic material is then mapped to the context of the real problem, the phenotype space, where it is possible to evaluate the quality of an individual. Recall that it was mentioned before that in biology, phenotype is a set of physical features which determines each individual's degree of adaptability.

This factor is considered the heart of an Evolutionary Algorithm, because it is a decisive factor between the success or failure of an algorithm and it should allow encoding all the possible solutions of a given problem. This representation relevance and dependence exists due to two reasons. First, it is important to notice that the way the variation operators work will depend upon the representation used and second, the same problem can be encoded with many representations. It is possible that a right representation choice precludes the algorithm from converging to a good solution and a right representation choice allows the algorithm to achieve a good solution. So, the representation choice should be done wisely.

Some common types of representations used are: binary, tree-based, grammar-based, graph-based and linear representation.

Fitness Function

John Holland defined the fitness of an individual as "its ability to survive and reproduce" [12]. This definition inspired by Darwinism will ultimately lead to a situation where only the fittest individuals will survive and will be able to reproduce. As they contain genes that favour their survival, these genes will be passed throughout generations, allowing the incoming individuals to better adapt to the environment [7]. This degree of adaptation is viewed as fitness, because it differentiates the best from the worst individuals according to their quality. It is a measure, which can be determined by a fitness evaluation function, assigned to each individual and it is an important component of an Evolutionary Algorithm.

Holland also mentioned the role of the fitness in providing robustness of an Evolutionary Algorithm, stating that if the same pattern of alleles appears in several individuals with an above-average fitness, this pattern will be disseminated to the rest of population in future generations. However, if the same pattern appears in a negligible number of individuals, it will disappear in subsequent generations, as a result of natural selection [12].

The importance of having a good fitness function in the sense that it has major influence on how the space of possible solutions explored. When formulating a method to measure the quality of a solution, one has to have in mind that a good fitness function may be hard to design because there is a drawback between precision and performance associated to computing a fitness function with a given complexity.

Population

In nature, a population is a set of individuals which somehow strive to survive and reproduce. Within the context of EC, a population holds the set of solutions which are used to perform the evolution process. In each so-called generation, the old population is transformed into a new population of individuals, through the application of variation operators. An evolutionary algorithm works over a set of individuals simultaneously to achieve hopefully better individuals and can allow multiple copies of the same individual, although this affects the diversity of the population.

When designing an evolutionary algorithm, one has to take into account the number of individuals used to compose the population. This number should be sufficiently large to promote the diversity of the population, but at the same time, not too large to avoid excessive computational burden that could compromise the evolutionary algorithm's capability to solve problem.

The number of individuals has also an indirect impact on the initialization methods used. The lower the number of individuals, the higher must be the concern about exploration ability, maintaining diversity and avoiding the local optima problem. In such cases, the population should be sampled not randomly, but somewhat premeditated [18], in order to optimise the coverage over the search space by the initial individuals.

Selection Operators

Selection operators aim to collect the set of prospective parents to be allowed to reproduce, based on the fitness of each individual. The set of offspring for the next generation will be created from the parents' set.

They take all the population into consideration according to each individual's fitness. This means that any individual is able to be part of the parents set. However, this process is probabilistic and the probability of being selected to the parents set depends on the fitness of each individual: the higher the fitness value, the higher is the probability to be selected. Hence, it is possible to prevent the premature convergence of the algorithm since it exists probability of choosing a below-average individual, avoiding a “greedy” approach.

Variation Operators

The role of the variation operators is to generate new solutions with the necessary diversity, creating new individuals from existing ones. This is done by imitating the reproduction mechanisms in nature. There are two main operators used which differ in the classic Evolutionary Algorithm in their arity (the number of operands necessary): recombination (or crossover) and mutation. However, other operators were proposed, such as transduction [19], conjugation [20, 21, 22] or transposition [23].

Recombination Also known as crossover, it is an operator that generally takes two so-called parents as input arguments and produces one or two descendants. Parts of the parents' genotypes exchanged between to produce the offspring. This process must be performed in such a way that the offspring hold a structurally valid genotype structurally. This operator is stochastic, meaning that is known that the parents will exchange information, but each genotype's portion is chosen through a non-deterministic process. The idea of using recombination is to hopefully combine the best portion of each parent and obtain an offspring with increased fitness that can better survive, it wants to exploit the search space. For this reason this operator is applied with a high occurrence probability (typically between 0.5 and 0.8 [9]).

Mutation It is an unary operator that creates a new individual by slightly modifying the input individual. The objective of this operator is to provide diversity by exploring the search space with random and unbiased changes over the individual. The principle behind mutation, in opposition to recombination, is to explore more unlikely areas of the search space, which might lead in the future to better solutions. Because it holds certain destructive (but interesting) properties, mutation is applied with a low occurrence probability (typically around 0.001 [9])).

Survivor selection mechanism

When the offspring are generated, the survivor selection mechanism has the role to select the individuals that will be part of the next generation.

There are two approaches to define the new population. The so-called generational approach performs **full replacement**, that is, the offspring will totally replace the parent population. On the other hand, a **partial replacement** approach does not perform a full replacement. Instead, in a simplified way, only the x best children will replace the x worst parents (assuming that the size of the population remains constant, which is the traditional approach, and $x < size_{pop}$).

Generalizing the survivors selection mechanism process and assuming that the size of the population remains constant, if the number of children is lower than the population, the children will replace a part of the population and the remaining older individuals will survive for the next generation. Otherwise, a full replacement can be done.

Inicialization method

Initialization methods in evolutionary algorithms are generally low-cost processing since it is usual to initialize the population randomly. However, in some particular cases (for example, in case of using lower population sizes as it was mentioned before) it can involve an extra computational effort. The initial solutions are created randomly, should be controlled in order to prevent the creation of invalid solutions.

Termination criteria

Evolutionary Algorithms run for a predefined length of time. Ideally, if one knows the optimal fitness level, when that fitness is reached by an individual, one can stop the algorithm and retrieve the solution.

However, Evolutionary Algorithms do not guarantee reachability to a global optima since they are stochastic. Even if a suboptimal solution is considered an acceptable one, using a suboptimal fitness value as a threshold or margin error to stop the algorithm increases the chances of ending the algorithm but it does not assure that it is terminated. To avoid the risk of getting stuck in the evolution process, other termination strategies were proposed to turn EC into a viable option [8]:

- The CPU time reaches its maximum level;
- The number of fitness evaluations performed reaches a limit;
- The number of generations reaches a limit;
- Fitness improvement remains under a given threshold value for a certain amount of time (Stagnation);

- Population diversity decreases under a given predefined threshold.

2.1.2 Genetic Programming

This EC method will be briefly described concerning the structures and operators used, since GP is the approach chosen to perform the evolution of images.

GP was first proposed by Cramer [24], and further extended by Koza [1], as a trial to evolve programs instead of simple structures. Koza argued that evolving programs could be used to solve problems without explicitly programming them. The main difference compared to other Machine Learning approaches (formal grammars, self-organizing maps, neural networks, decision trees, etc.) remains in the use of a generalized structure while other Machine Learning approaches use GP use computer programs special structures to solve a problem.

GP evolves stochastically individuals into new individuals through generations, trying to obtain the best individual possible although there are no guarantees. The main distinguishing feature of GP is the type of individual used to evolve: a program. From now on, the terms individuals, programs and candidate solution will be used interchangeably.

GP's main advantage is the construction of increasingly more complex structures, which causes the growth of the genotype length. The structure is not restricted to a fixed length. However, this growth can occur in an uncontrolled way, without increasing the fitness. This phenomenon is denominated **bloat** [25].

Representation in GP

Programs can be represented through abstract syntax trees, they are more intuitive and there is a lot of work performed in GP using trees [1, 16]. Although tree representation is the most widely used in GP, linear [26] and graph [27, 28] representations can also be used. As an example, Figure 2.2 represents the program $max(x + x, x + 3 * y)$. Notice in this program that the variables x and y, and the constant 3 are the leaves of the tree. This subset is designated in GP as the **terminal set**. On the other hand, the operators $+$ and $*$ correspond to the internal nodes of the tree. This set is designated in GP as the **function set**. Together, the function set and the terminal set form the **primitive set**. Table 2.1 shows an example of a primitive set used in GP.

As indicated in the example in table 2.1, both the function set and the terminal set can have several types of data. For example, functions can range from a simple

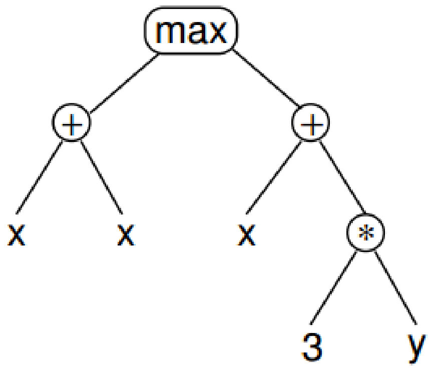


Figure 2.2: Example of a program using tree representation [17].

Table 2.1: Example of a primitive set that can be used to solve a possible GP problem [17]

Function Set	
Kind of Primitive	Example(s)
Arithmetic	+,*,/
Mathematical	sin, cos, exp
Boolean	AND, OR, NOT
Conditional	IF-THEN-ELSE
Looping	FOR, REPEAT
Terminal Set	
Kind of Primitive	Example(s)
Variables	x, y
Constant values	3, 0.45
0-arity functions	rand(), go_left()

add operation to a loop. Terminals have constants, variables and functions which do not take any arguments.

Irrespective of the primitive used, Koza defined two main properties that must be satisfied in order to have a GP algorithm to run properly: sufficiency and closure.

Sufficiency enunciates that the primitive set should be sufficient to express a solution for the problem. The problem is that, in general, it is hard to know what is the correct primitive set to use. It is only possible to know in cases which theoretically or empirically one knows in advance the right primitive set to express a solution. Following a trial-and-error approach, it can happen that the primitive set is insufficient, and in that case, one cannot obtain the optimal solution, but an approximate one. On the other hand, if unnecessary primitives are added, the algorithm does not slow down too much, but it can create unexpected biases.

The other property defined by Koza was **closure**, to assure type consistency and evaluation safety. Basically, type consistency means that any subtree can be used as an argument of any function which requires arguments. This fact implies that the data type used remains consistent, and if there is a possible problem with incompatible types, conversion mechanisms can be provided. For example, the expression $AND(1, -1)$ generates a function which accepts two boolean operands, but instead this expression has two integer operands. A possible solution consists in defining a non-negative number as $TRUE$ and a negative as $FALSE$. However, these mechanisms can introduce unexpected biases in the problem. Evaluation safety guarantees that functions are handled for cases in which they fail at runtime, such as the division of a number by zero. To solve this problem, protected versions of these functions are created or individuals which fail in the evaluation phase are strongly penalized in their fitness.

Initialization methods

Initializing individuals in GP is typically a random process. It consists in creating trees using functions and terminals as nodes. Before explaining the main methods used to initialize the population, there are two properties used to analyse an individual. The **depth** of a tree is the number of edges that need to be traversed to reach the furthest terminal node. Concerning a metric related with nodes, the **size** of a tree is the number of nodes in a tree.

All the methods that will be described below use the depth of a tree as a reference in the initialization process. The **full method** consists in building trees with the same depth for every path that connects the root to the leaves (a predefined depth that should not be exceeded). Figure 2.3 illustrates the generation process step-by-step ($t = time$) using the full method with a maximum depth of 2. Nodes from the function set are added until the defined depth is reached (in $t = 1, 2$ and 5). After that, only nodes from the terminal set can be used to fill the remaining of the tree ($t = 3, 4, 6, 7$).

Although this method fills the tree with nodes until it is full, one could immediately deduce that all the the programs generated will have the same size. However, this situation only happens if all the nodes from the function set present in all the trees have the same arity. But even considering an optimistic scenario where the function set has functions with different arities, full method generates trees with a quite limited range, which does not favours diversity.

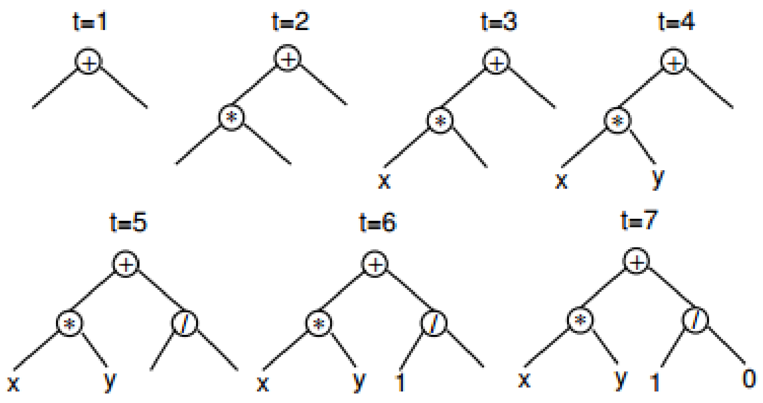


Figure 2.3: Iterative scheme of full method initialization strategy [17].

A different method, known as **grow method** tries to overcome this shortcoming. The main difference compared to full method is that both nodes from the function and terminal sets can be used to fill the tree. This means that, despite having a maximum depth limit to the generated tree, not every path from the root to the leaves must have the same depth. Figure 2.4 illustrates the generation process step-by-step using the grow method with a maximum depth of 2.

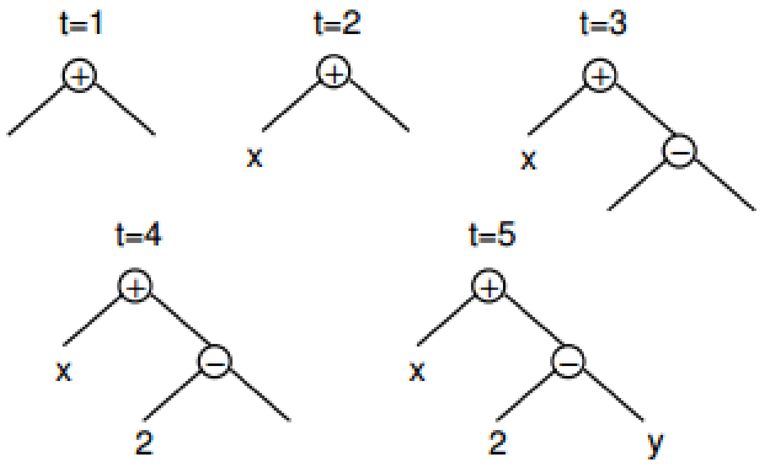


Figure 2.4: Iterative scheme of grow method initialization strategy [17].

Notice that with this method, in the moment $t = 2$ a terminal node is used to fill is chosen instead of a function one. Automatically, this node will be a leaf and the depth will be lower than the predefined.

The grow method is very sensible in what concerns the size of the terminal set and function set. If the number of terminals is significantly larger than the number of terminals, grow method will generate short trees, regardless the tree depth defined. On the other hand, if the number of functions is significantly higher than the number of terminals, grow method will generate very similar trees when compared with the full method.

Despite of creating different tree types (analysing both tree depth and size), grow method and full method are not able to provide substantial variety in the created trees. To overcome this problem, Koza proposed a combination of those methods, known as **Ramped Half-and-half**. This method uses several depths. For each depth half of the individuals are created with the grow method and the remaining half are created with the full method. Table 2.2 summarizes the creation process of Ramped Half-and-Half, choosing a population size of 20 and a number of depths that ranges from 2 to 6 with individuals I_i in which i identifies an individual. For example, I_1 is an individual created with the grow method with a depth of 2.

Table 2.2: Description of initialization of 20 individuals and depth varying from 2 to 6, created with Ramped Half-and-half method.

Method	d=2		d=3		d=4		d=5		d=6	
Grow Creation	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}
Full Creation	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}	I_{20}

Ramped Half-and-Half is the most widely used method to initialize populations in GP because it provides the necessary structural diversity to the population. However, there are some problems associated with this method [17] and sometimes, one has to take into account that if some properties of the problem are known, and if that happens, a non-random initialization may improve the performance of the GP algorithm.

Fitness functions

Fitness functions are meant to provide information about which regions of search space are good or not. In GP, individuals are programs and in order evaluate them, each program must be executed to know the fitness measure.

In GP, fitness measures can be expressed in several ways. It is possible to measure fitness by minimizing the error between an expected output and the real output or maximizing the accuracy in classification problems where one has to recognise patterns. A fitness metric can also be built around the amount of resources used to accomplish a predefined task (e.g. time), or taking into account the payoff resultant from a set of actions taken by a game-playing program or even measuring the compliance of a structure with a predefined criteria.

In order to evaluate each individual/program, an interpretation is performed in such a way that each node is only executed when the value of their arguments is known. This is done by traversing similarly to a depth-first search from root to leaves and executing a node only when all of its children were already executed. This evaluation process is schematically described in figure 2.5.

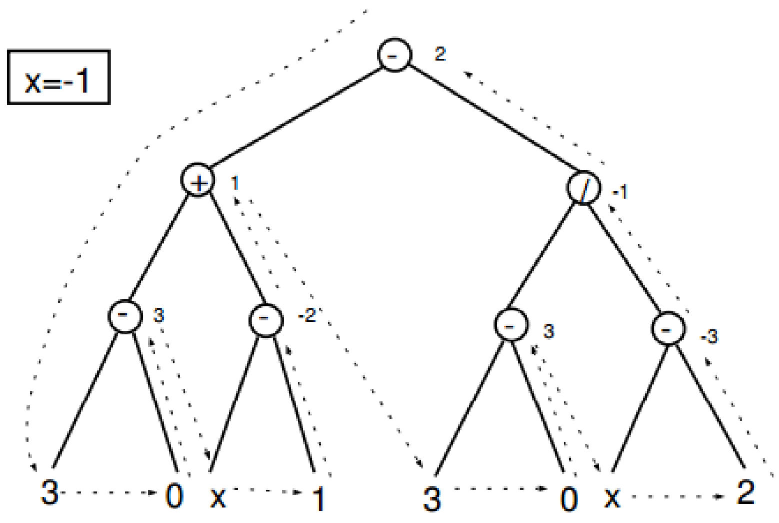


Figure 2.5: Example of a tree evaluation to compute fitness measure [17].

Selection Operators

As in EC in general, GP uses a selection mechanism based on the fitness of each individual. There are several ways to provide selection [29]. However, only the tournament selection is going to be focused because it is the most common alternative chosen within GP's scope [17] and it is the selection method chosen for this work.

An important aspect of selection is that whatever is the selection mechanism chosen, it has to be selective enough (promote selection pressure) to accelerate the convergence and fitness improvement, but at the same time not too selective and consequently compromising population diversity by exploring local optima.

Tournament selection chooses a parent by randomly retrieving a predefined number of candidates from the current population and then selecting the best of them. Given that a crossover needs two parents, two tournament selections will be needed to perform a crossover. Selection pressure can be regulated by adjusting the number of candidates retrieved. For example, if the number of candidates is 1, it is the equivalent to a random selection. On the other hand, if the number of candidates is the same of the size of the population, the best individual will be chosen. A reasonable number of candidates, capable of coping with the drawback mentioned before, will be able to provide the rise of the fitness by choosing the best individuals to breed and at the same time allow some average-quality individuals to breed if they are the best from the candidates chosen to tournament selection, avoiding the premature convergence of the algorithm. The most common values chosen for tournament selection are between 2 and 10.

Variation Operators

In tree-based GP, both recombination and mutation operators are adapted to work with tree structures. As in other EC approaches, GP uses these operators with a given probability. Typically, recombination is used with high probability values (around 90% [9]) whereas mutation is applied with very low probability values (roughly 1% [9]).

In some cases, when the sum of recombination and mutation probability (denoted by p) does not pass 100%, another operator is introduced, known as **reproduction**, which will act with a probability of $(1 - p)$. This operator only clones the selected individuals.

Recombination

In GP, the most common type of recombination used is **subtree crossover**. Given two parents, this methods selects in each parent a crossover point, in a probabilistic way. Then, copies of the parents are created to avoid their disrupt. These copies are used to perform the exchange of subtrees. The crossover point in the first parent will be the root of the subtree used to replace the crossover point of the second parent. The second offspring is generated similarly, by replacing the selected subtree in the first parent with the selected subtree of the second parent. This process is illustrated in figure 2.6.

Given the structural properties of trees, if an uniform distribution was used to select crossover points, they would have higher probabilities of being chosen in nodes with higher depth (in a complete binary tree the number of leaves is higher than the number of internal nodes). This fact would imply that crossover would exchange less information. To prevent this situation to happen, Koza suggested to attribute 90% of selection probability to internal nodes and 10% to external nodes.

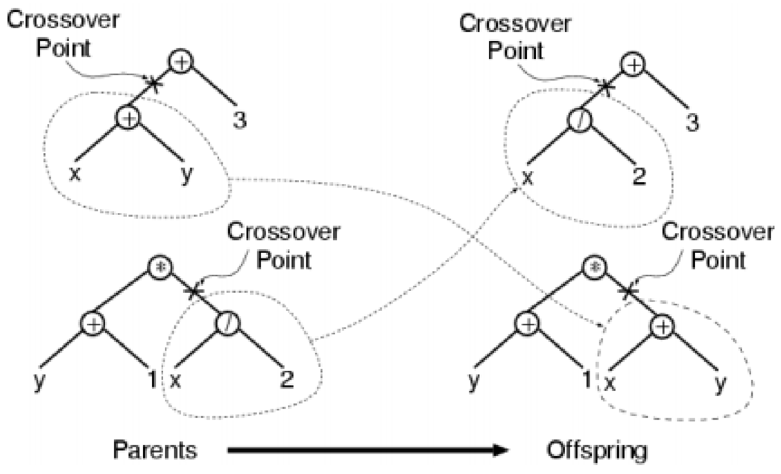


Figure 2.6: Description of subtree crossover process [30].

Mutation

The most common form of mutation in GP is **subtree mutation**, also known as "headless chicken crossover" [31].

In summary, a new subtree is generated and this new subtree will replace a randomly selected subtree within the individual, like in subtree crossover. The difference is that there are not two individuals involved but one instead. Besides, new genotype information is introduced through a new tree since there are no second individual to exchange genetic information with. This process is illustrated in figure 2.7.

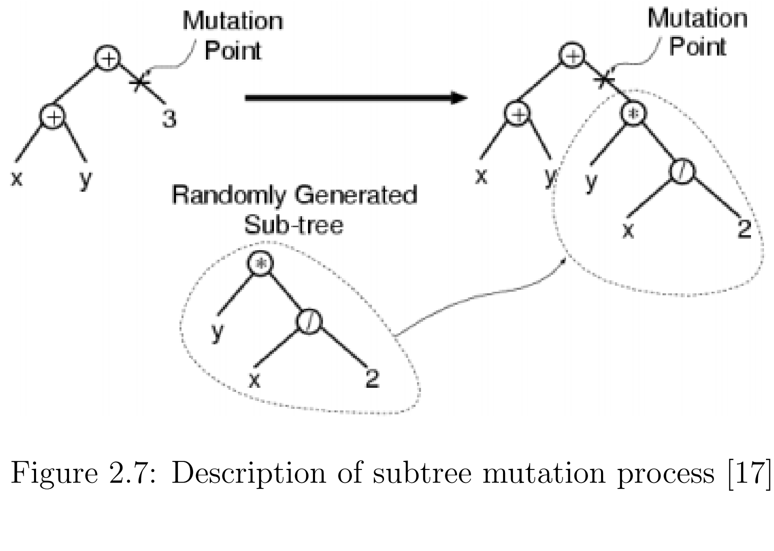


Figure 2.7: Description of subtree mutation process [17].

2.1.3 Multi Objective Evolutionary Algorithms

Real-world problems do not usually have a single global optimal solution and they are not subjected to a single objective. Instead, they have multiple and equally good optimal solutions. This fact occurs because if a solution is analysed according to several often conflicting objectives, there is not any solution better than all of the others over all objectives. Consequently if one wants realistic solution, the problem has to be solved in a more complex way.

These so-called multiobjective optimization problems, tend to be trickier to solve because their objectives may conflict each other, which means that an improvement over one objective usually comes at the price of other(s) objective(s) degrading.

A formal definition of a multiobjective optimization problem is described in equation 2.1:

$$\begin{aligned} \underset{Y}{\text{maximize}} \quad & Y = \mathrm{f}(x) = (f_1(x), ..., f_n(x)) \\ \text{subject to} \quad & x = (x_1, ..., x_m) \in X, \\ & y = (y_1, ..., y_n) \in Y; \end{aligned} \tag{2.1}$$

where one wants to maximize a set of functions that maps m parameters into n objectives. In equation 2.1, x denotes the decision vector, X denotes the parameter space, y represents the objective vector and Y the objective space.

The set of solutions of this type of problem called as Pareto optimal, is composed of solutions that cannot be improved in any objective without degrading another.

In order to understand the Pareto optimal concept, it is necessary to introduce dominance and non-dominance. One says that solution a dominates solution b ($a \succ b$) according to equation 2.2.

$$\begin{aligned} & \forall i \in \{1, ..., n\} : f_i(a) \geq f_i(b) \wedge \\ & \exists j \in \{1, ..., n\} : f_j(a) > f_j(b), \\ & a, b \in X. \end{aligned} \tag{2.2}$$

On the other hand, when a decision vector is not dominated by any of the others decision vectors, it is a non-dominated solution. Pareto optimal set, or Pareto front, contains all the non-dominated solutions.

One possible way to tackle these types of problems consists on using Multi Objective Evolutionary Algorithms (MOEA's). These approaches have been preferred over non-population-based methods during the last years due to the characteristics of Evolutionary approaches, as they can explore several solutions in the search space simultaneously, in a single run.

In MOEA's, evolution must favour the appearance of hopefully better solutions taking into account an entire set of objectives rather than a single one. However, as it was said before, most of the times it is not possible to select an overall best solution.

Based on the overviews in [32, 33], three main techniques of using evolutionary algorithms to evolve optimal solutions according to several objectives are described and explained.

Plain aggregation techniques When applying the "survival of the fittest" principle in a simple evolutionary algorithm, scalarization is necessary in order to select the best individuals by comparing their fitness. Plain aggregation techniques merge all fitness values (each value corresponding to each objective) into one single value that is used for selection purposes. Consequently, the selection process is made in the exact same way as a single objective evolutionary algorithm. This can be seen as an advantage because the general evolution process of the evolutionary algorithm is not changed, it only needs an additional combining function to convert the entire set of values into a single one.

However, the capability of finding an optimal solution depends on the combining function used and formulating such a function is a hard task and requires knowledge about the problem domain. The effort of finding an optimal solution through this technique is worsened by running the algorithm several times, with different combining functions. In the literature two popular approaches can be highlighted, (1) the weighted-sum approach [34], which consists in obtaining an aggregated fitness value as a result of a linear combination between single fitness values and weights associated to each single objective and (2) the target vector optimization [35], in which the algorithm minimizes the distances between the objective values and a goal vector that contains the designed goal of each objective.

Population-based non-Pareto techniques Another possible way to address multiobjective optimization using evolutionary algorithms is to influence the evolution direction during reproduction phase. Here, the parents are selected by considering the different objectives, i.e. some parents are selected according to one objective, while others are selected according to another. This way, the concept of best individual will not get blurred by an aggregation and non-commensurable goals are treated separately. Within this scope, two works can be highlighted, (1) the choose of individuals according to a priority list of objectives selected by the user [36] and (2) the control of each objective's importance using an associated weight value which varies according to the average fitness considering each single objective [37].

Pareto techniques Pareto approaches benefit from the fact that all the non dominated solutions are treated in the same way in terms of quality evaluation. Therefore, there is no risk of valuing an objective more than the others.

Two ways of applying Pareto approaches are reported. The first one, introduced in [38], consists in ranking the individuals according to how many other individuals dominate it, where non-dominated individuals are ranked above the ones which are dominated. The more individuals dominate a given individual, the lower will be its fitness. Following this idea, the Non-dominated Sorting Genetic Algorithm-II (NSGA-II), was proposed [39].

Pareto dominance at the tournament level was introduced in [40]. In this case, the pareto domination criterion is applied in a binary tournament where the individual that dominates the other wins. In case of neither individual dominating the other, the winner is determined by analysing the existence of individuals within their neighbourhood. The exploration of these techniques intensified and became popular and widely used within the scope of multiobjective optimization

2.2 Maintaining diversity of the population

In biology, diversity plays an important role by providing several ways to survive in the same environment. The higher the diversity, the higher will be the survival probability of a given species by the mean of creation of innovative and novel behaviours.

In EC, the same phenomenon occurs while one is seeking the best solution. Maintaining the diversity of a population because during the evolutionary run the algorithm tends to converge to a set of very similar solutions in terms of fitness. If this convergence happens in a suboptimal search space zone, the task of finding the global optima becomes harder. Concerning the diversity topic, the initialization method is important because it is expected to create individuals that will cover the search space with the lowest bias possible. If individuals assume a vast range of solutions, then it is less probable that the algorithm converges to a local peak (the local optima problem). Population size is important too, because the larger the number of individuals, the higher will be the coverage of the search space, minimizing the risk of leaving unexplored areas of the search space that may contain a global optima.

However, increasing the number of individuals will cause the algorithm to be more time-consuming and resource-demanding.

Several techniques exist to promote diversity [41, 42], taking into account either genotypic measures [43] or phenotypic measures [44]. This work will focus on novelty search and fitness sharing, using measures at the phenotype level, because these techniques match with this work objectives.

2.2.1 Novelty Search

When one addresses the fitness functions' topic, it is almost implicit that one is talking about objective-based fitness functions since the work of Holland and others was disseminated throughout the literature about EC. The idea behind this type of fitness functions consists in having a specific target, an objective in mind, which is used as a reference to evaluate how close a candidate solution is to the optimal solution.

One can characterize these fitness functions as **convergent**, in the sense that they are typically used for optimization (maximization or minimization) problems and the candidate solutions are gradually evolved to obtain increasingly closer solutions to the optimal solution (in the best case scenario the optimal solution itself). This evolution occurs over the different generations of the Evolutionary Algorithm and it is expected to improve the quality of the population on average, as the algorithm proceeds through generations. After a finite number of runs, if the fitness is precise and expresses correctly the desired improvement, one expects to have the algorithm exploring more promising regions as the individuals are located nearby the optimal solution.

Despite the reported success and dissemination of these objective-based functions, some authors argue that objective-based approaches narrow the potential of an Evolutionary Algorithm [45]. The reason for this claim lay on the fact of the effectiveness of an objective function in an Evolutionary Algorithm depends on the problem's characteristics, namely if it is a deceptive problem or not. The concept of deception was introduced by Goldberg [38] when he realised that the above-average individuals were not globally competitive when testing the algorithms with a set of deceptive problems created before [46, 47, 48]. Since then, several studies addressed the deceptive phenomenon.[49, 50]. With objective functions, ambitious goals have higher difficulty of being pursued, as fitness landscapes become more complex because of the difficulty of formulating appropriate fitness functions. Thus, evolution can guide individuals to dead ends in the search space.

While some authors focused on studying the deception itself and its effects on an Evolutionary Algorithm's performance, others studied a way to mitigate deception.

What if the best way to reach an objective could be attained by not searching for the objective itself? This is the idea behind the novelty mechanisms in Evolutionary Computation. Instead of looking for something specific, towards a fixed objective, the Evolutionary Algorithm evolves according to a so-called novelty degree instead of the traditional fitness. The more distinctive is the solution, the higher will be its novelty value. Thus, if one tries to reward the uniqueness of each individual, the Evolutionary Algorithm will strive for different solutions along the generations. In the context of Art and Computational Creativity, Saunders developed a computational model that consisted of multiple agents looking for novel images generated by them through GP [51] inspired by "The Law of Novelty", a thought settled by Martindale in his work known as "The Clockwork Muse" [52]. Their novelty metric was based on a subjective definition of interestingness which states that what makes something interesting depends upon two aspects: unexpectedness and actionability [53]. To compute the novelty value of each individual, a Self-Organizing Map (SOM) (described in section 2.2.3) was used.

A different novelty search approach was proposed by Lehman and Stanley [45, 54]. Instead of using a SOM to obtain a novelty measure directly, each pair of individuals has a dissimilarity value associated. Thus, the final novelty measure from an individual x can be obtained by adding all the dissimilarity values between the individual x and its neighbours μ_i in which i identify an individual different from x. As this process is very time-consuming, the k-nearest neighbours distance relative to x are used to compute the novelty measure. Equation 2.3 resumes the way to calculate novelty measure ρ given an individual x.

$$\rho(x) = \frac{1}{k}\sum_{i=1}^{k} dist(x, \mu_i) \tag{2.3}$$

where k is a fixed parameter determined experimentally and μ_i is the ith-nearest neighbour concerning a domain-dependent distance measure $dist$ between x and μ_i. In this context, Liapis et al. proposed a distance measure used to analyse the dissimilarity of two images using Auto-encoders (described in section 2.2.3) [55]. Then, this distance measure was applied on Novelty Search algorithms.

This approach could bring some problems exploring the search space, because maximizing explicitly novelty using exclusively equation 2.3 does not prevent a "backtracking" behaviour, where the algorithm is exploring already explored solutions, like a random walk. To avoid this behaviour and transform novelty search into a viable alternative, a novel archive was used to store the most relevant (novel) individuals found until the current population. To allow storage of the most relevant individuals only, a threshold ρ_{min} was defined and individuals were only added if they reached that minimum value. Thus, the k-nearest neighbours take into account both individuals from the current population and the novel archive. Figure 2.8 summarizes the evolution process of an algorithm which uses novelty search.

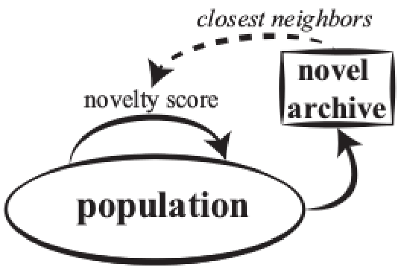
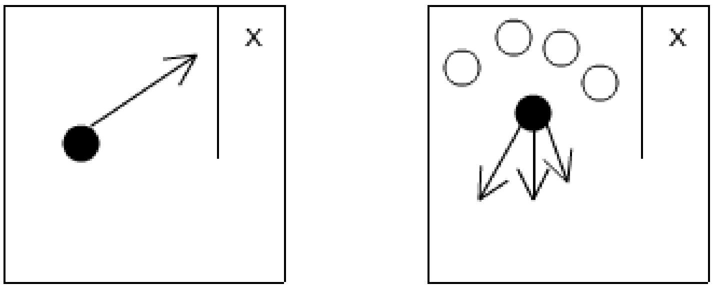


Figure 2.8: Novelty search process along generations [56].

Distinguishing visually how the implicit process of both fitness-based search and novelty search is performed is intuitive. Figure 2.9 illustrates which ares of the search space are more likely to be explored, using a maze as an example and assuming that one wants to minimize the distance from the individual to the final maze position. In figure 2.9a, the candidate solution (represented by a filled circle) will strive to get as close as possible to the optimal solution (represented by an "X") in a eager way. After hitting the wall, the candidate will more likely stay in similar positions as it falls into the trap induced by a deceptive problem. In figure 2.9b the candidate solution is always trying to be as unique as possible. If one considers the set of individuals represented by white circles, the candidate solution will try to "run away" from the other individuals, and eventually traversing the right path in order to get the final position in the end.

In spite of outperforming the traditional objective-based fitness functions, novelty search has a **divergent** nature, because it is not directed to any particular goal. It overcomes the deceptiveness of a problem, but there is no particular focus in solving the problem explicitly because there are no guiding references about how close are the candidates solutions relative to the optimal solution. Novelty search per se is not suited to constrained optimization problems.

Lehman and Stanley worked around this problem by presenting the Minimal Criteria Novelty Search (MCNS) [58]. The purpose of this approach was to explore



(a) Fitness-based Search. (b) Novelty Search.

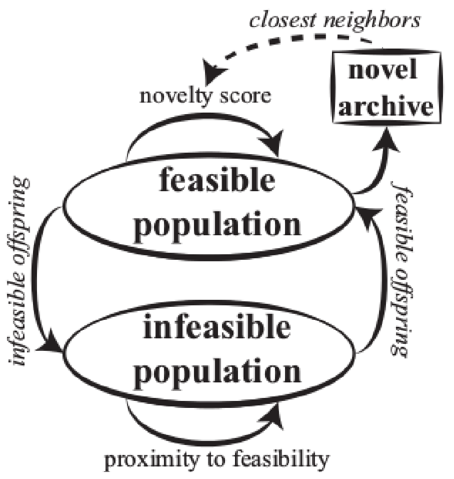
Figure 2.9: The rewarding mechanisms in Fitness-based search and Novelty Search [57].

novelty search while enforcing individuals to have a given requirement. This solution was capable of offering a different guiding mechanism, focusing on exploring novelty in valid solutions (or feasible solutions) only. MCNS extends the default novelty search by determining if the candidate solution is a feasible solution or not in the evaluation phase. If the candidate is a feasible individual, novelty search operates as normal, but if the candidate is an infeasible individual, the novelty value will be zero.

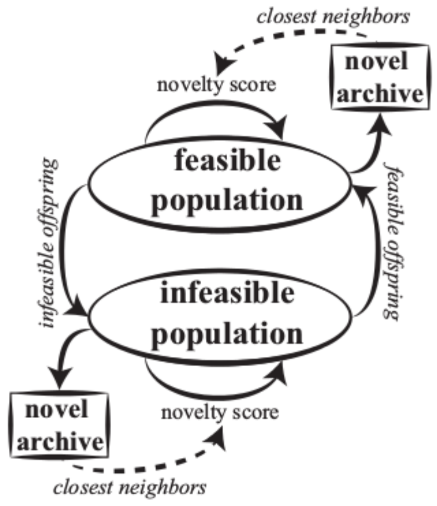
The default version has difficulty to cope with large search spaces and MCNS prunes infeasible regions, and this is why the latter achieves better results than the default novelty search. However, MCNS performs a random walk while no feasible individual is found, because there is a trial to evolve individuals which have the same fitness value.

Liapis et al. proposed two different approaches of constrained novelty search [59], using Feasible-Infeasible Two-Population Algorithm (FI-2pop GA)[60] as an inspiration. FI-2pop GA operates with two different sets with independent heuristics to compute fitness values: the set of infeasible individuals and the set of feasible individuals. In each generation the population is divided into these sets and breeding is performed only between individuals of the same set. Nevertheless, it is possible that the mating of two feasible individuals generates an infeasible individual and the mating of two infeasible individuals generates a feasible individual. FI-2pop GA allows the migration of the offspring in those cases, which promotes diversity in both populations.

Novelty Search was introduced with FI-2pop in two different forms. The first one is known as FINS and performs an objective-based search in the infeasible set of individuals and a novelty search in the feasible set. The idea is to first obtain valid individuals, and once they are obtained, one starts to evolve them in order to retrieve



(a) FINS algorithm.



(b) FI2NS algorithm.

Figure 2.10: Representation of FINS and FI2NS evolution processes along generations [56].

a vast range of feasible solutions. In order to know how close infeasible individuals are to being part of the feasible set, a border which separates the two sets must be defined. The closer infeasible individuals are to the border, the higher will be their fitness.

However, determining a good border is a hard task, and to avoid that problem a different flavour of FI-2pop was created. This flavour is known as FI2NS and it differs from the first in the sense that both sets, infeasible and feasible, uses novelty search, which saves the work of designing the border.

These algorithms were employed in DeLeNoX system [55]. In this work, both use equation 2.3 to compute novelty for each individual, using a specific distance measure between two phenotypes. This metric uses auto-encoders to detect relevant features from images and these features will be used to assess the uniqueness of an individual (auto-encoders are further described in section 2.2.3).

Comparing these two methods, the authors concluded that FI2NS was able to generate more diverse content but it had difficulty in maintaining an acceptable number of feasible individuals. On the other hand, FINS had better results on highly constrained search spaces although it generated less diverse content.

Figure 2.10a illustrates how evolution is processed along generations using FI-2pop GA with Novelty Search in the infeasible set of individuals (FINS) while figure 2.10b illustrates evolution process of FI-2pop GA with Novelty Search in both sets of individuals (FI2NS).

The problem of novelty search being explored in smaller search spaces as it suits better these kind of problems is referred also by Cuccu and Gomez [61]. Their idea starts from this motivation and consists in doing two evaluations per individual: one in terms of fitness-based search and the other concerning novelty search. These two values are combined explicitly using equation 2.4.

$$score(x) = (1 - \alpha) \cdot \overline{fit}(x) + \alpha \cdot \overline{nov}(x) \tag{2.4}$$

The result, $score(x)$ is obtained by performing a weighted sum between fitness calculated from the fitness-based search($\overline{fit}(x)$) and the value calculated from novelty search ($\overline{nov}(x)$). The weight factor is denoted by α. In the end, the final value $score(x)$ is the value actually used to guide the evolution process. It is also relevant to notice that both $\overline{fit}(x)$ and $\overline{nov}(x)$ are normalized values according to the actual population and not simple raw fitnesses. Their normalization is calculated using equation 2.5,

$$\overline{fit}(x) = \frac{fit(x) - fit_{min}}{fit_{max} - fit_{min}}, \quad \overline{nov}(x) = \frac{nov(x) - nov_{min}}{nov_{max} - nov_{min}}, \tag{2.5}$$

where $fit(x)$ and $nov(x)$ are respectively, the raw fitness and novelty degree of individual x, normalized according to their maximum and minimum value in the current generation. Maximum values for fitness and novelty are respectively denoted by fit_{max} and nov_{max} while their minimum values are denoted by fit_{min} and nov_{min}.

2.2.2 Fitness Sharing

Multimodal problems are characterized by having multiple optima instead of a single global optima. In this situation, a traditional genetic algorithm often converges to one of those optima instead of identifying all of them. This happens because the selection operator induces a strong pressure towards convergence in spite of the existence of the mutation operator. As this operator is not able to provide the necessary diversity to multimodal problems, other techniques must be used.

Fitness sharing was proposed by Holland [12] and further improved by Goldberg and Richardson [62]. It is a niching method used in genetic algorithms to search in the neighbourhood of several peaks in parallel and is based on the idea of speciation and competition among the individuals of each niche who strive to obtain resources to survive. The higher the number of individuals within a niche, the lower will be the survival probability of all individuals inside that niche because they will need to share the same resources. This encourages niches with fewer individuals to survive.

In EC, this idea can be implemented by worsening the raw fitness of individual i (f_i) according to its niche count m_i, based on the number of individuals belonging to that niche and their proximity to individual i. The niche count is calculated as it is represented in equation 2.6.

$$m_i = \sum_{j=1}^{N} Sh(d_{i,j}), \tag{2.6}$$

in which N represents the size of the population, $d_{i,j}$ represents the distance between individuals i and j and Sh is a sharing function which calculates how similar two individuals are, according to equation 2.7.

$$Sh(d_{i,j}) = \begin{cases} 1 - (\frac{d_{i,j}}{\sigma})^{\alpha} & \text{if } d_{i,j} \leq \sigma \\ 0 & \text{if } d_{i,j} > \sigma. \end{cases} \tag{2.7}$$

in which α is a constant parameter that is normally set to 1 and it regulates the shape of the sharing function. In turn, σ is a parameter that denotes the boundary which separates the niche radius from the rest. Only individuals within this boundary will be considered competing within the niche.

Thus, the final fitness of an individual i ($\hat{f}_i$) can be obtained using equation 2.8.

$$\hat{f}_i = \frac{f(i)}{m(i)} \tag{2.8}$$

Despite of favouring exploration of search regions which were not explored yet, fitness sharing has shortcomings too. One of them is related with the σ parameter, as it is defined in advance and, generally, one does not have information about the search space. Besides, the computation of the sharing fitness for all individuals in all generations is a time-consuming task. To overcome this problem, Oei, Goldberg and Chang proposed a method to update fitnesses continuously. Fitness sharing was applied alongside with a binary tournament selection scheme until all the parents were selected to generate the offspring. Furthermore, Yin and Germay suggested the employment of a clustering algorithm over the population and the sharing fitness would be computed taking into account only the individuals belonging to a given niche instead of the entire population.

2.2.3 Measuring diversity

In a trial to distinguish techniques which aims to promote diversity in the population from metrics used to measure diversity, section 2.2.1 and 2.2.2 address the promotion of diversity only mentioning which metrics were used with the techniques employed. This helps the reader keeping track of the subject addressed in each subsection while receiving only the most relevant information. Thus, this subsection addresses how diversity metrics were employed in a deeper way. Recall that until now, two metrics were identified: one that uses auto-encoders and other which uses SOMs to associate an uniqueness degree to each individual.

Auto-encoders

Auto-encoders [63] are a type of artificial neural networks which can be used for dimensionality reduction or data compression purposes. In the context of imagery, auto-encoders can be useful to represent images with a more abstract and compact representation. For the sake of simplicity and contextualization within this work, only the architecture of the auto-encoder employed in DeLeNoX will be explained in detail.

An auto-encoder is a feedforward network and it has an input layer, an hidden layer and an output layer. Although it can be trained using backpropagation algorithm [64], its learning method is placed in the category of unsupervised learning. An example of an auto-encoder is presented in figure 2.11.

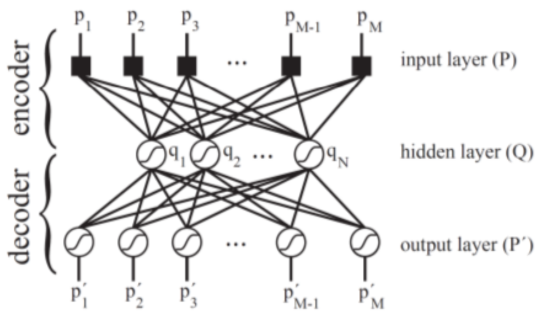


Figure 2.11: An example of an Auto-Encoder [55].

In the backpropagation algorithm error is computed using equation 2.9a, where r is the real output and y is the expected output. With auto-encoders, to compute the error, the input value x is used instead of y, as it is possible to notice in equation

2.9b. This happens because, conceptually, input and output layers aim to represent the same space and the auto-encoder will be trained to learn the identity function.

$$E = \frac{1}{2} \cdot (r - y)^2 \tag{2.9a}$$

$$E = \frac{1}{2} \cdot (r - x)^2 \tag{2.9b}$$

Basically, an auto-encoder with the input space P and m dimensions will represent the same data in a lower dimensional space Q with n dimensions ($n < m$) by performing an encoding using the parametric function described in equation 2.10a, where W is the weight matrix, b is the bias from the input layer and sig is the sigmoidal function used as activation. Then, a decoding is performed by mapping back Q into the original space P', where data is reconstructed with a small reconstruction error using equation 2.10b, where $W^\intercal$ is the weight matrix transposed and b' the bias from the hidden layer. As the number of dimensions in the hidden space is lower than the number of dimensions in the original space, one can consider an auto-encoder as a lossy method to compress data.

$$Q = sig(W \cdot P + b) \tag{2.10a}$$

$$P' = sig(W^\intercal \cdot Q + b') \tag{2.10b}$$

In the context of imagery, an auto-encoder was used to retain the most significant features from an image in a higher level. These features were used to compute the novelty degree of an individual. Based on equation 2.3 from section 2.2.1, equation 2.11 was defined,

$$\rho(x) = \frac{1}{k} \sum_{i=1}^{k} \sqrt{\sum_{n=0}^{N} [q_n(x) - q_n(\mu_i)]^2} \tag{2.11}$$

where x is the individual being analysed, k is the number of neighbours, N is the number of features in the hidden layer, q_n identifies a neuron from the hidden layer and μ_i identifies one of the neighbours of x used to compute novelty. In this equation it is possible to notice that an euclidean distance between individuals is used, after retrieving high-level features for them, in the hidden layer.

Self Organizing Maps

Self Organizing Maps (SOMs) [65] are non-traditional neural networks. When compared with the most traditional ones (perceptron, ADALINE, Multilayer Feedforward Networks, etc.) they have structural differences and they learn how to classify data without supervision. Instead of working with traditional layers, SOMs have an input layer with n neurons and a lattice of neurons, each with a vector of n weights associated because each input node has a connection to every node in the lattice, as it is represented in figure 2.12.

The learning algorithm is built as it follows. Weights vectors are randomly initialized and then, examples from the training data are presented to the lattice. For each example presented, the algorithm determines which node is the most similar to the example presented. This node is designated as the Best Matching Unit (BMU), represented in black in figure 2.12. This node can be determined using a simple distance measure such as the Euclidean distance.

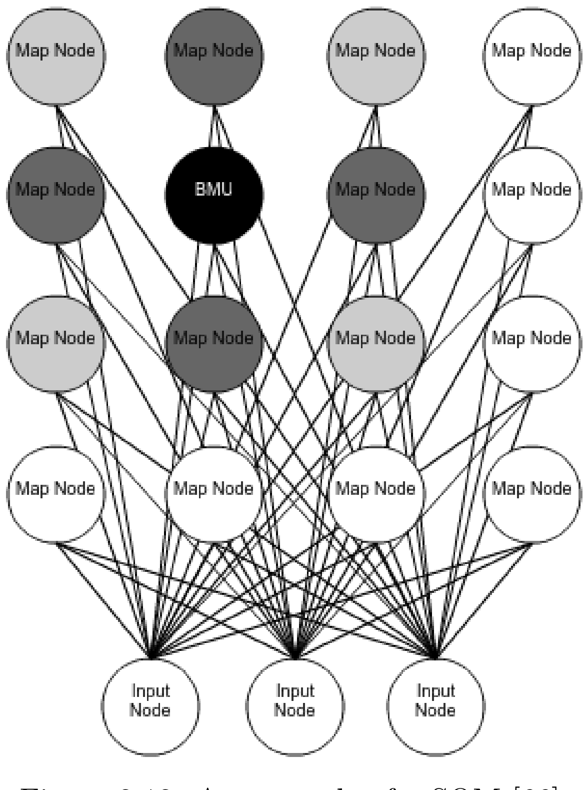


Figure 2.12: An example of a SOM [66].

In the next phase, weights vectors are going to be updated. Once again, this perspective differs from the most traditional ones. In SOMs, nodes are organised in a lattice because each node has its own coordinates values, making possible to compute distance between nodes. This distance is used to select the nodes that will suffer a weight adjustment. Basically, there is a neighbourhood parameter that marks the region of nodes that need to be adjusted in each iteration. This neighbourhood region will decay over time according to equation 2.12.

$$\sigma(t) = \sigma_0 \cdot e^{-\frac{t}{\lambda}}, t \in \mathbb{N} \tag{2.12}$$

where $\sigma(t)$ denotes the neighbourhood radius value in the time instant t and λ is a constant.

Weight adjustments are performed over each node within the lattice using equation 2.13

$$W_{t+1} = W_t + \Theta_t \cdot \alpha_t \cdot (I_t - W_t), t \in \mathbb{N} \tag{2.13}$$

where I_t is the input vector in time instant t, W_t is the weight vector of a given node in time instant t, α_t denotes the learning rate in time instant t and Θ_t represents the influence of a node's distance from the BMU.

As it is possible to notice, learning rate and distance influence are parameters which varies over time. Learning rate is a parameter which decays over time according to equation 2.14

$$\alpha_t = \alpha_0 \cdot e^{-\frac{t}{\lambda}}, t \in \mathbb{N} \tag{2.14}$$

where α_0 is the initial learning rate and the remaining variable were already mentioned. Distance influence is computed using equation 2.15.

$$\Theta_t = e^{-\frac{dist^2}{2\sigma_t^2}}, t \in \mathbb{N} \tag{2.15}$$

where *dist* denotes the distance (it can be the euclidean distance mentioned before) between the node and the BMU and σ_t is the neighbourhood radius value computed using equation 2.12.

In the context of novelty search, a SOM was used to compute a novelty degree over each candidate that is going to be classified [51]. For this purpose, a lattice size must be defined, and each node within the lattice denotes a possible category or type determined in the train phase. When one is presenting a given example to a SOM and the BMU is determined, it is assumed that the BMU is the category which better fits the characteristics evidenced by the example. From this moment, one reasonable approach to assess novelty is to compute the difference between the category defined by the BMU and the example presented. This can be done through the calculation of the classification error of a given example.

2.3 Evolutionary Art

Evolutionary art is a research field which overlaps concepts related with art, nature and science. It "allows the artist to generate complex computer artwork without them needing to delve into the actual programming used"[67]. The artistic dimension of this field is present on the final products performed by the machine, whether they are music, imagery, video, etc. Science assumes its role on Evolutionary Art (Computer Science in particular) by handing over the creator role gradually from the human to the machine. Finally, Nature has its influence by inspiring the artistic production process based on the biologic roots left by Darwin.

Metaphorically speaking, one can consider the evolutionary art process in the following way. Given a director (the environment, in biology) which leads a team of artists (individuals), they will strive to solve a problem by presenting their solutions. Then, these solutions are evaluated by the director, who decides the ones who are the best according to some given criteria (fitness). After receiving feedback from the director, the best solutions are selected (natural selection) and new solutions will be produced based on the selected ones (reproduction).

A computer can create a similar process using EC techniques. They can be applied in the form of Genetic Algorithms albeit the most common EC flavour used in the context of imagery (which is within the scope of this work) is Genetic Programming.

Using Genetic Algorithms it is possible to represent an object as a set of parameters. Lets say for example that one wants to evolve cartoon faces. Thus, one can define one parameter as the radius of an eye, another as the thickness of a mouth and others parameters can be defined. This whole set of parameters is the genotype and according to the value of each gene, the corresponding image will be produced with a given appearance, creating the phenotype. A population of these cartoon faces can be generated by adjusting the parameter values. Examples of phenotypes with different gene values are presented in figure 2.13.

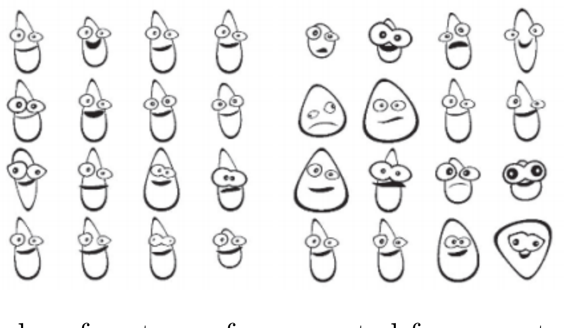
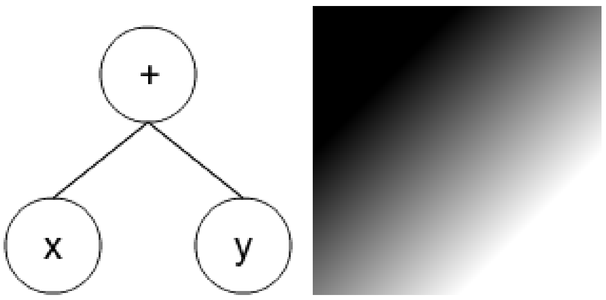


Figure 2.13: Examples of cartoons faces created from a set of parameters [68].



(a) Tree representation of expression `x+y`. (b) Image generated from expression `x+y`.

Figure 2.14: Genotype and phenotype using genetic programming approach to generate images [4].

Concerning Tree-based Genetic Programming, it is possible to represent an image using an expression (it can have mathematical operators, logic operators, decision operators, etc.) which is represented by a tree (the genotype). In this case the image appearance will be influenced by the set of operators used, how mapping between genotype and phenotype is performed, as well as Genetic Programming specific parameters. An example of a tree (figure 2.14a) and its respective image (figure 2.14b) are presented on figure 2.14.

Evolutionary Art is divided in two major branches in the remainder of this section. Section 2.3.1 addresses interactive evolutionary systems, where the evolution user-guided. On the other hand, section 2.3.2 addresses automatic evolutionary systems, which are systems capable of evolving artistic productions autonomously. All of these subsections comprise only imagery topic, which is the only one within the scope of this work.

2.3.1 Interactive Evolutionary Systems

Interactive Evolutionary approaches in the context of Evolutionary Art have been used often, mainly because it is hard to define a good fitness function which is able to control evolution process autonomously. In these kind of systems, the human plays the role of evaluator, deciding which individuals will be used to breed.

The first system to appear within this category was The Blind Watchmaker, created by Richard Dawkins [69]. He used genetic algorithms to evolve what he calls "biomorphs" (vaguely animal-like shapes) by associating a determined gene to a given characteristic of the biomorph (position, angle, length). He chose to evolve these 2D images to show the impact of applying small changes over a significant amount of time and prove that in a narrow time window, new entities appear by chance. However, the final product is the result of "cumulative selection", which is not a pure random process. Examples of biomorphs are shown in figure 2.15.

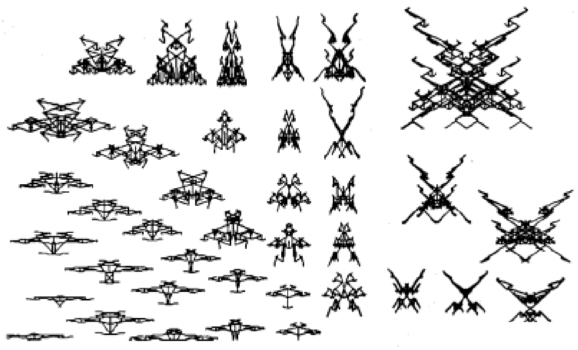


Figure 2.15: Examples of biomorphs created by Dawkins [69].

Following the steps of Dawkins, Sims created what is today the most popular approach to evolutionary art, the expression-based approach [2]. Using GP approach to evolve images, textures and animations, Sims pointed out that fixed-length genomes would limit the search space and it was important to remove this boundary from the evolution process to allow the creation of more complex structures. Genotypes were LISP expressions that were organised according to tree-based genetic programming: each internal node was a LISP function and each external node was a constant or variable. Everytime a symbolic tree was executed, a new image was produced. The user was able to guide the evolution process like in The Blind Watchmaker, evaluating each image resultant from the corresponding expression. This evaluation would enable the mating and mutation of symbolic expressions using more probably those who generated images that met users taste. After this phase, a new generation of individuals was created and the whole process was repeated.

Many authors started to get inspiration in Sims work to develop their own applications. Those were the cases, for example, of Rooke, which created a wider set of functions, including fractals [70], Unemi, which created a multi-field interface[1] to gather important features from different populations evolved independently [71], Machado [72] who identified the problem of memory usage storing many images as a knowledge base and proposed a method of automatic seeding to solve that problem, and Hart, whose work focused particularly on color and form of the evolved images [73].

[1]The author defined field as a population of individuals that can be visualized and selected by the user

Expression-based systems are in fact very powerful and theoretically, they have been proven to be able to generate any image [4]. Nevertheless, in practice, generated images tend to be abstract and mathematical based. Besides, the search space explored will depend on system parameters such as the primitive set, genetic operators, genotype-phenotype mapping, etc.

Other techniques of Evolutionary Art were also explored, such as the line/shape based approach. Concerning this approach, Ellie Baker used genetic algorithms to evolve drawings as a set of lines. In this case the genotype was not composed by symbolic expressions [74]. Instead, a set of strokes was used, where each one had a set of parameters that described how each stroke should be performed. In this work, two operating modes were created. One of them consisted in evolving line drawings from a random initial set of drawings according to the user criteria. The authors obtained butterfly forms and face shapes, despite of the difficulty of this achievement. These results were the motivation of a second operating mode where the user performed a preliminary sketch and then submitted it to a interactive evolution process, which allowed to collect variations of interesting images.

Despite of the more common use of Interactive Evolutionary approaches, several limitations have been pointed out [75]:

User fatigue It is the biggest problem identified by researchers. In interactive approaches, evolution depends on the human user. After a while, repeating the same task all over again can induce monotony and fatigue, with the increase of evaluation error.

Lack of consistency in evaluation Art is a subjective field, the decision of liking or not some image depends on the user context and tastes. It is possible that a user can vote positively for some image, and then change his mind, if the same image appears again after some generations. Consequently, evolution can be guided in an inconsistent way.

Novelty is valued instead of quality Evolutionary algorithms tend to converge to some optima after a number of generations. In this case, it is common to have a set of very similar images which can even be pleasant the human user. However, the lack of diversity in those images will cause human eyes to highlight a possible "outlier" that can be not so pleasant and value it more than the other images.

Need to search under poor conditions Keeping in mind the human users limitations, pleasant images must be obtained within a reasonable number of generations (avoiding fatigue) and with a low number of individuals (more individuals means more evaluations and more memorizations to be done in order to compare images). Under these conditions, evolutionary algorithms tend to have a bad performance in terms of the quality of the produced images.

Slow process Evolving images where evaluation is conducted by the human user is not just a tiresome task, is a very slow process which could be improved if evaluation was performed by the machine.

In order to solve some of these problems, some applications run their evolution software with voting/server systems [76, 77, 78]. This option is particularly useful to avoid human fatigue, as the evaluation can be distributed through several users at the same time.

However, this is a controversial option from an artistic point of view if images are not generated always by the same user, because the artist does not review himself in the final product [72].

A different solution proposed to avoid interactive approach problems consisted in creating partial interactive evolving systems. They save user effort while trying to maintain the quality of generated images. Concerning this field, Machado et al. [79] suggested a mechanism which allowed the user to choose either if he wanted to perform interactive mode, taking the responsibility of guiding evolution, or if he wanted to perform automatic evolution for a given number of generations. In case of automatic approach, evolution was guided by a formula that related several complexity estimations.

Nevertheless, one of the most relevant challenges of Evolutionary Art comprises the full automation of evolution process. Theoretically, it has capability to solve all the problems mentioned above, but other issues are introduced because the quality of the images generated tend to be considerably lower. How can a machine learn to model human preferences in a context of multiple users with mixed choices? Which criteria should be used in order to produce pleasant images? Researchers who study automatic evolutionary art are trying to answer to these questions.

2.3.2 Automatic Evolutionary Systems

Automatic Evolutionary approaches differs from interactive ones in one key aspect: the evaluation is performed by the machine, relieving the user from performing a time-consuming task. Three main approaches used to automate fitness assignment of artworks were identified: hardwired fitness functions, machine learning approaches and co-evolutionary approaches.

Hardwired fitness functions are made with a specific purpose. Their formula has some theory or idea behind the function. Thus, they require more domain knowledge, and their generalization cannot be achieved. Within the scope of imagery, Machado and Cardoso built a fitness function which aimed to generate pleasant images from an aesthetic point of view [4]. Their theory behind the fitness function was that aesthetic visual values depends on two factors. The first is Image Complexity (IC), which is desired to be maximized, in order to capture the most attention possible from the human eyes. This metric was computed using Root Mean Square Error (RMSE) and *jpeg* compression. The higher the compression achieved, the less aesthetic value it would have, because in this metaphor, aesthetic is about causing some reaction on the human brain through some kind of surprise, change or anomaly. However, they considered that complex images do not imply a difficult way to build them, giving the example of fractal images, which are generated in a simpler way. Following this idea, the second factor is Processing Complexity (PC) which is desired to be minimized. PC was computed through fractal compression.

Greenfield developed fitness functions based on color segmentation performed over images [80]. According to the different regions areas and perimeters, several fitness functions were developed.

Ralph and Ross proposed a different measure, the bell curve gradient [81]. They described its computation in three steps. The first step consists in computing the image's color gradient, for each RGB channel. Then, the stimulus of an image is calculated for each pixel, using the Euclidean distance of the three channels. The second step constructs a normal distribution, computing the mean and standard deviation from the previous computed responses distribution. The final step computes the deviation from normality (DFN), where 0 means a perfect fit to a normal distribution. They stated that for example, painting, photographs and graphic designs have different values of mean, standard deviation and DFN, which allows evolution to be conducted towards different kinds of imagery. These three values were used after to perform multi-objective optimization.

Machine Learning approaches are more likely to be generalized, as they do not require specific knowledge. At most, they need to know users tastes in order to predict them.

The work of Baluja et. al is the most noticeable using Machine Learning approaches to evolve images autonomously albeit they had some inconclusive results. They used an Artificial Neural Network (ANN) to evolve images and estimate aesthetic preferences of the user. The ANN was trained with a set of images with size 48x48 px. These images were retrieved during an initial period of interactive evolution. After that, the ANN tried to predict the user preference, which could be a value ranging from 0.0 to 1.0, with increments of 0.1. Higher values mean more pleasant images. A lot of future work and interrogations remained, concerning the tuning of population size and the ANN architecture for example. The work of Correia et. al is also within this scope, but it will be explained in section 2.3.3.

Co-evolutionary approaches assume that an interaction between different types of populations exists, where their fitnesses are assigned according to a degree of "supremacy" over others populations or "reciprocity", depending on one is tackling competing or cooperating environments, respectively.

Greenfield explored this approach by evolving filters and images following a parasite-host relation [82], where filters play the role of parasites and images play the role of hosts. The rationale was that the interestingness of an image is influenced by our vision, the information absorved and consequently, the interpretation of that information in our brain. An interesting image is one which is recognized by our brain as visually significant. This idea of absorption was applied through convolution filters in the original image in the following way: if the convolved image is different than the original one they are more interesting than the ones where both convolved and the original images are similar. Fitnesses for both images and filters were computed based on this principle, the more different an image is, the more its fitness is valued. Filters fitnesses have the complement value, considering the fitness upper bound 100.

Saunders and Gero also proposed a co-evolutionary environment where the fitness was the interestingness metric mentioned in section 2.2.1 [51]. Basically, they created an agent society, where each agent could evolve its own artworks, and if they exceeded a given threshold, they were shared to other agents. These agents could incorporate received images in their population, to induce some diversity, and even share them in a public domain, labeled with the respective author. Therefore, not only each agent is evolving its own creations (with the help of the the remaining agents), but also an agent gives credit to another agent if it considers the artwork creative and meaningful. This particular work combines also the co-evolutionary approach with a machine learning approach, as the novelty value is computes with a SOM.

2.3.3 Figurative Expression Based Evolutionary Art

When Karl Sims worked on his first images generated by his evolutionary engine, one of them became famous because he could not save the expression that would be able to generate the image in figure 2.16. The expression was lost and he was unable to reproduce that image again using an expression-based approach.

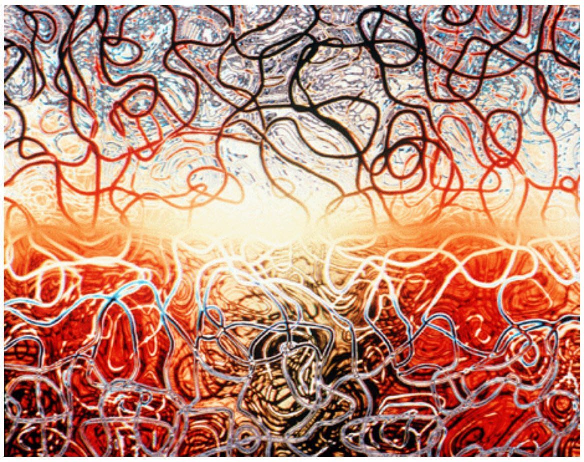


Figure 2.16: The "lost" image in Sims work [2].

In his future work, Sims mentioned that it would be interesting to evolve symbolic expressions using a specific goal image. John McCormack identified this problem as an open problem in the field of Evolutionary Art and described it as "finding a needle in a haystack" [3]. Within a search space so vast, it is very hard to apart interesting images from the rest automatically, taking into account that evolutionary techniques will tend to generate portions of a perfect image with some fuzziness mixed and obtaining the whole perfect image is a scenario with very low probability. He also affirmed that using an expression-based system to represent images, despite of creating a wide range of possibilities, is a limited approach because they tend to have a "certain class" [83], which difficulties the task of finding a symbolic expression to represent a given figurative image. However, Machado and Cardoso have proved that NEvAr system, an expression-based one, was able to theoretically generate any image [4].

Within this scope, Paola and Gabora [84] tried to use a GP approach to evolve towards Charles Darwin portrait, but their results were disappointing. In fact it was hard for a human to detect any resemblance between the generated images and the goal image.

This problem can be interpreted as a symbolic regression problem where one tries to find the model that correctly maps a set of coordinates to the final image. This

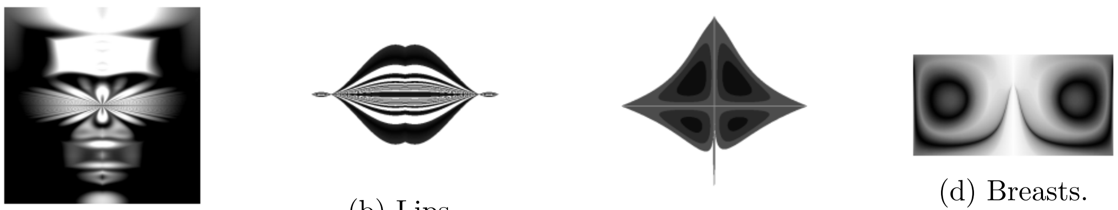

(a) Faces. (b) Lips. (c) Leaves. (d) Breasts.

Figure 2.17: Figures evolved using an evolutionary engine with an object detector [87].

problem can change its difficulty whether one tries to evolve any expression or a compact expression.

Despite of being an interesting problem to solve in terms of compression of code length, evolving symbolic expressions to approximate from a goal image are useless from an artistic point of view because there is no creativity nor novelty in performing such task.

However such a problem can be transformed in order to become more interesting in terms of creativity. Instead of measuring resemblance from a solution to the goal image, one can generate several candidate solutions using GP (the creator) and measure them using an object classifier (the critic) which evaluates solutions according to its resemblance to a predefined object, without boundaries on its characteristics. Romero et al. [85] proposed a framework which could be applied to several domains (they gave the example of music and image) where proposed artworks were submitted to a artificial art critic (AAC). The AAC converted artworks to an internal representation in order to be evaluated by an Evaluator Module which was modular and could adapt to perform different evaluation tasks. In the end, artworks would be assessed in order to identify the corresponding author or style.

Since that moment, Correia et. al worked on the idea of having an external object detector and an evolutionary engine. This engine would evolve images according to the object classifier trained. Thus, it is theoretically possible to evolve any object if there is a classifier capable to detect it. Efforts have been made to prove this, through the evolution of faces (figure 2.17a) [5, 86], lips (figure 2.17b), leaves (figure 2.17c) and breasts (figure 2.17d) [87], using a cascade classifier proposed by Viola et. al [6] and trained with Haar features (further described in section 3.2). This work is the big inspiration of this thesis.

Chapter 3

Evolving Figurative Images

This chapter addresses the evolution of figurative images without the user's help. In order to fulfill this goal, a tool known as geNeral purpOse expRession Based Evolutionary aRt Tool (norBErT) was developed, inspired on the work of Machado and Romero [85]. This chapter intends to prove that their proposal is able to evolve images evocative of different objects and this is done through a set of experiments that aims to assess the tool's ability to create different images that resemble different objects.

The first three sections tackle important decisions regarding the application of their approach on norBErT. Section 3.1 explains the approach used to evolve figurative images. It also describes implementation details such as frameworks used and changes made in order to meet work requirements. Section 3.2 describes the role of an object classifier within the approach, its operation mode and which features were used to train it. Section 3.3 details how the classifier contributed to computing the fitness values. Consequently, the designed fitness function is presented and explained. Section 3.4 contains all the information about the tests performed. It is divided in section 3.4.1 which contains the parameters set used in the experimentation, section 3.4.2, that shows relevant results obtained, namely interesting images generated and the study of fitness modification over the generations and section 3.4.3 which analyses the results showed in the previous section. Finally, section 3.5 summarizes all the subjects approached in this section in a more compact form the findings about evolving figurative images.

3.1 Evolutionary Engine

Machado and Romero proposed a general purpose approach where it was possible to evolve figurative images towards any object, depending on a chosen object classifier

[85]. In [5, 86, 87] the same approach was used to evolve faces, face silhouettes, leaves and lips. This approach was selected for this work and it is presented in figure 3.1.

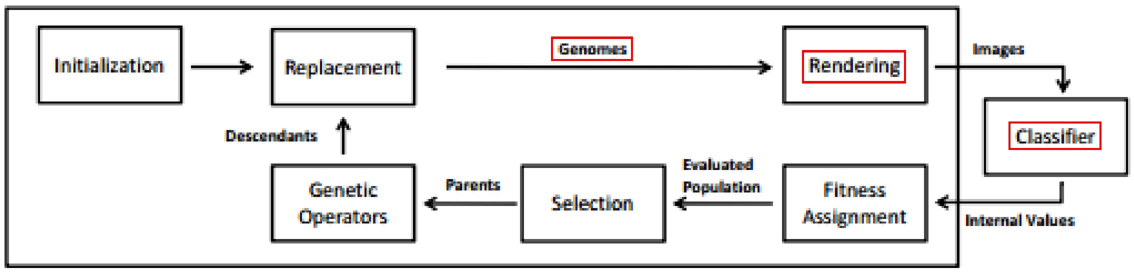


Figure 3.1: Architecture used for the experimentation work[5]. Red marks symbolize major changes performed.

The approach presented in figure 3.1 has several similarities when compared to the generic evolutionary algorithm presented in figure 2.1. A given number of individuals is initialized to fill a population. Then, the population is evaluated in a slightly different way in relation to the generic algorithm. In EC, fitness is obtained after evaluating a given phenotype, which are images in this particular problem. To create these images, one needs to gather all the genotypes from the population and convert them into images. That is what rendering module is responsible to do. Then, an object classifier will evaluate the generated images in order to detect whether the image contains an object or not. This step is very important within the whole approach because the classifier produces internal values that are used to compute the fitness function and, therefore, assign a fitness value to each individual. The remaining steps are the same of the generic evolutionary algorithm as selection and mating operators are applied to the population. Afterwards, the offspring will replace the old population in order to start the same process all over again.

In order to develop the chosen approach, a Java-based evolutionary computation software, known as ECJ [88], was used. This choice was made mainly for two reasons. First, the system is developed in Java, providing portability, and second, the author had already some past experience working with ECJ.

ECJ provides several EC features to build and run evolutionary algorithms. In order to adapt the generic evolutionary algorithm and recreate the approach referenced, major changes were made in components marked in red in figure 3.1. The remainder of this section will describe in detail changes performed in each component.

Genomes In expression-based art, GP is the most used EC approach, and this work is not an exception. Individuals' genomes are trees, composed by a list of primitives hierarchically connected. A genome of an individual is influenced by the primitive set used to solve a problem. Table 3.1 describes primitives used to construct GP trees. To construct this set, primitives used in [5] were assembled with other operators introduced just in case they were needed.

Table 3.1: Primitive set used in the proposed architecture.

	Function Set	
Kind of Primitive	Primitive	Arity
Trigonometric	sin, cos, tan	1
Other mathematical	sign, neg, abs, exp, sqrt, log	1
Arithmetic	+,-,*,/	2
Other mathematical	max, min, mdist, mod	2
Boolean	AND, OR, XOR	2
Conditional	IF	3
Distortion	warp	3

Terminal Set	
Kind of Primitive	Primitive
Variable	x, y
Single Ephemeral Random Constants (S-ERC)	e.g: 0.2
RGB Ephemeral Random Constants (RGB-ERC),	e.g: (0.2, -1, 0.5)

Trigonometric operators and arithmetic operators performs the regular mathematical operations, needing 1 and 2 arguments, respectively. Operator `sign` is a function which returns 1 if the argument is positive, -1 if the argument is negative, otherwise, returns 0. Operator `neg` inverts the signal of the argument, `abs` returns the absolute value, `exp` performs the natural exponential function(e^x), `sqrt` performs the square root and `log` the logarithm. Other mathematical functions that takes 2 arguments such as `max` and `min` returns the maximum or minimum between two arguments, respectively, `mdist` returns the mean distance between two arguments and `mod` performs the integer division using a dividend and a divisor. Boolean functions perform bitwise operators over two arguments (in this case, the operators `AND`, `OR` and `XOR` are used) and the conditional operator `if` accepts three arguments, applied in the following form: if a then b else c, where a, b and c denote arguments. The distortion operator `warp` is a bit more complex. It gathers an image (*img*), an abscissa *coordx* and an ordinate *coordy*, returning a distorted version of that image (*img*′). Both *coordx* and *coordy* are obtained from two pixel values of two different images. For example, if one is calculating the first pixel of *img*′, *coordx* and *coordy* are obtained by accessing the values of the first pixel of two different images. Then, the first pixel of img' will assume the value of $img(coordx, coordy)$.

Concerning the terminal set, x and y are variables that represent abscissas or ordinates of the image, S-ERC are normal constants and RGB-ERC are special constants which provides the capability of generating colorful images, overcoming the restriction of being limited to grayscale images.

Image rendering In GP, the evaluation of an individual is performed by interpreting the tree (its genotype) to obtain a value. This value can be used to measure fitness in terms of error, by calculating the difference between a predefined value and the value obtained after interpreting the tree.

In this problem, an individual requires requires an evaluation per each image pixel, in which each final value will be the color of the respective pixel. For example, if one wants to render an image with size 10x10 px from a genotype, the tree must be interpreted 100 times, once for each pixel. Within this context, variables x and y from the terminal set presented in table 3.1 assume a critical role providing diversity to the generated images. If these terminals did not exist, the image generated would have the same color in all of its pixels.

In order to use x and y variables, a virtual Cartesian coordinate system is used to map an image into it. This system ranges from -1 to 1 both in x axis and y axis. The mapping consists in creating the number of pixels necessary into the intervals $x \in [-1; 1]$ and $y \in [-1; 1]$, according to a predefined resolution. This means that the rendering module can render images of any resolution. Then, values of x and y are determined for each pixel in order to be input in the individual's tree. Figure 3.2 summarizes this process using an image with size 10x10 px.

Using figure 3.2 as a reference, the first tree interpretation is done for the first pixel (the one in the upper left corner), using values $x = -1$ and $y = 1$. After computing the value of the first pixel, the process is repeated for the second pixel (next to the first pixel, in the same row) with values $x = -0.8$ and $y = 1$. The third iteration is processed with $x = -0.6$ and $y = 1$ and the process continues iteratively until all the pixels are interpreted.

After interpreting the tree and obtaining a pixel value, this output is converted to a value within the RGB scale (between 0 and 255) for each channel, according to equation 3.1.

$$treeoutput = \frac{treeinput + maxDomain}{2 \cdot maxDomain} \cdot 255 \tag{3.1}$$

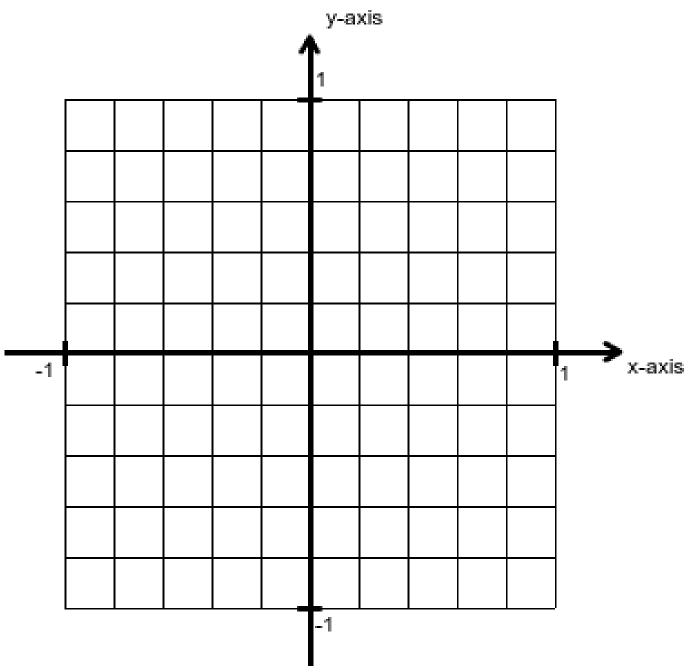


Figure 3.2: Mapping an image 10x10 px into the coordinates.

where $maxDomain = 1$ because it is the maximum possible value in the axis presented in figure 3.2 and the input is a value within the interval $[-1; 1]$. Values outside of this interval are truncated.

Classifier Observing figure 3.1, it can be seen that the classifier component is external to the EC environment. This gives the idea of modularity and possibility of using a classifier trained to recognise any object. In terms of implementation, the classifier is considered external to the engine because it runs outside of this environment. For this purpose, OpenCV API [89] was used to detect whether images contain the object for which the classifier was trained. OpenCV was compiled into its wrapped Java version and integrated in the evaluation phase of ECJ, where a fitness value is assigned to each image entry using the internal values of the classifier.

3.2 Classifiers

For this work, the cascade classifier proposed by Viola et. al [6] was chosen to detect the existence of objects within an image. The cascade classifier has been used before in expression-based figurative imagery [5, 86, 87], it is able to classify examples fast and it had a considerable impact regarding academic contribution. The training and detection processes using these object classifiers were already implemented on OpenCV.

In order to detect whether an object exists or not, an object detection algorithm is applied to choose which parts of an image will be submitted to the cascade classifier. This allows to locate possible objects that exist within an image. The detection process can be resumed in these steps (steps gathered from [86]):

1. Define a window of size w (20x20).
2. Define a scale factor s greater than 1. For instance, 1.2 means that the window will be enlarged by 20%.
3. Define W and H as the width and height of the input image.
4. From (0,0) to (W,H) define a sub-window with a starting size of w for calculation.
5. For each of these sub-windows apply the cascade classifier.
6. Apply the scale factor s to the window size w and repeat 5 until window size exceeds the image in at least one dimension.

The cascade classifier is composed by stages, each stage containing a so-called strong classifier. Given that an image can produce many sub-windows to the cascade classifier, it is important to give more attention to sub-windows that actually contain the desired object than others which contain wasteful information. To do so, each stage aims to discard non-objects and retain sub-windows that may contain objects.

The rationale is the following, simpler classifiers are built in the beginning of the cascade in order to discard unrelated sub-windows. The further a sub-window progresses through the cascade, the more attention it deserves. This means that more advanced stages will have more complex classifiers to analyse whether the sub-window should advance to the next stage or should be discarded. If a sub-window passes through all the stages, the interpretation is that the cascade classifier detects an object. The cascade architecture described is presented in figure 3.3.

Each strong classifier is trained with the AdaBoost algorithm, which selects the best features instead of using all of them and settles that the final classifier is a composition of weak classifiers. In figure 3.3, a strong classifier is defined for each stage, with a set of features that will be used to evaluate a sub-window, in order to decide whether the sub-window passes to the next stage or it is discarded. The weak

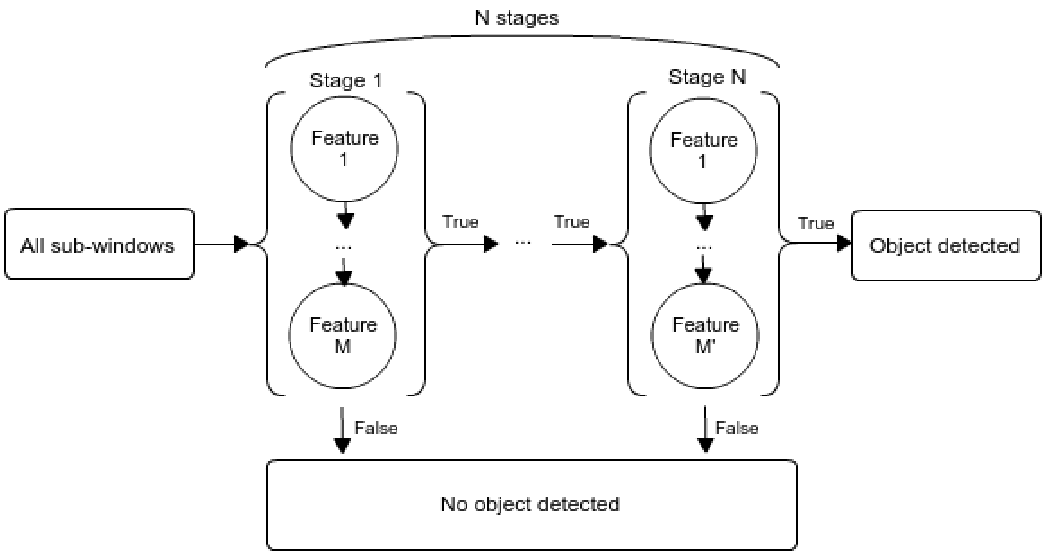


Figure 3.3: Structure of the cascade classifier used to detect objects [87].

classifiers are decision trees that use a single feature to divide positive and negative examples with the least classification error possible.

Three types of features are implemented in OpenCV and expected to be used in the cascade classifier: Haar features, Local Binary Patterns features (LBP) and Histogram Oriented Gradients features (HOG). However, HOG features were discarded because they had less documentation associated in OpenCV and they turned out to be very time-consuming during the training and detection phases.

Haar Haar features work by defining rectangular areas of black and white zones. After applying a given Haar feature to a subwindow, the value of this feature is computed as the difference between the sum of the pixels within the black and the white zone. Then, this value is compared to a threshold calculated during the training process. If this value is higher than the threshold, the feature exists on the image. An original set of Haar features was proposed by by Viola et al. , further extended by Lienhart et al. [90]. This extended set presented in figure 3.4 is used in this MSc study.

LBP LBP features capture more information about the image than Haar features. However, they contain redundant information. Assuming grayscale images, the simple LBP version introduced by Ojala et al. [91] is applied in a 3x3 window where px_c holds the value of the pixel in the center. Then, each one of the neighbours $px_i, i = (1, 2, ..., 8)$ is compared against p_c using a threshold function,

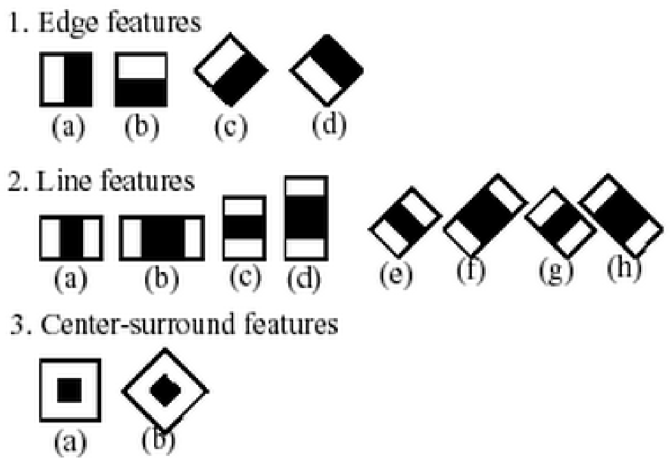


Figure 3.4: Extended Haar features [90].

specified in equation 3.2

$$value = \sum_{i=0}^{8} 2^i \cdot sign(px_i - px_c) \tag{3.2}$$

where *sign* is defined as in equation 3.3.

$$sign(i) = \begin{cases} 1 & \text{if } i \geq 0 \\ 0 & \text{else.} \end{cases} \tag{3.3}$$

Figure 3.5 shows how the threshold function is applied on the original pixel values in order to obtain the comparison outputs. The LBP window is applied to every possible location within the image sub-window and all the output values are gathered in order to compute the histogram corresponding to the image sub-window. Since each computed value has 8 bits, the histogram will have 256 different bins which will be used in the cascade classifier for classification purposes.

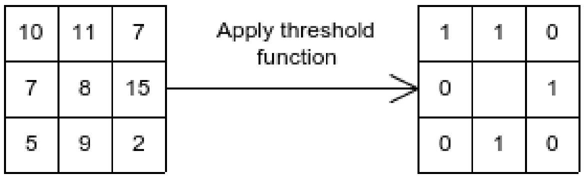


Figure 3.5: LBP feature applied to a 3x3 block.

Further information about the application of these features in cascade classifiers can be found in [92].

3.3 Fitness Functions

Using a classifier to detect whether an object exists or not is a binary classification problem. Using the binary value as the fitness guiding function would not be practical because this fitness value could be no indication of how close a candidate solution is to a perfect solution and the search would essentially become a random search.

A fitness function appropriate for evolving figurative images should value images that resemble the object that is being evolved and penalized images that do not. From this perspective, cascade classifiers can be a significant help because the more stages an image pass, the more similar the object is expected to be with the desired object. This means also that, the more stages the classifier has, the more accurate the evolution process can be. Following this idea, a fitness function was formulated by Correia and Machado by accessing internal values of the classifier [86] as presented in equation 3.4

$$fitness(x) = \sum_{i}^{cstages_x} beststagedifference_x(i) \cdot i + cstages_x \cdot 10 \tag{3.4}$$

where $cstages_x$ denotes the number of stages that a given example x has passed and $beststagedifference_x$ denotes the highest difference achieved on the ith stage between a value attained by example x and the threshold necessary to pass a stage. Besides valuing images with more stages passed, equation 3.4 has another variable, $beststagedifference_x$, which settles that images with values near the threshold are more doubtful to resemble the desired objects. On the other hand, images that greatly passes the threshold have more probability to resemble the object. Therefore, these images will have higher fitness.

This fitness function was integrated in ECJ in order to evolve objects according to the values returned by a given object classifier.

3.4 Experimentation

The experimentation performed was characterized by three steps. Since it was intended to develop new classifiers, there was a need to create datasets which allowed classifiers to learn how to detect the desired objects. This study covers four objects: faces, face silhouettes, flowers and leaves. Then, datasets were used to train new classifiers. For this purpose, OpenCV provided an executable which trained classifiers according to a set of predefined parameters. Finally, trained classifiers were integrated in the approach mentioned in section 3.1, to provide internal values needed to compute fitness during the execution of the evolutionary algorithm.

Section 3.4.1 describes the preparation of each step mentioned above, showing the relevant parameters sets. Section 3.4.2 presents examples of results obtained for each object covered by this study and their respective graphics regarding the evolution process. Finally, section 3.4.3 analyses the results obtained.

3.4.1 Setup

In order to train classifiers, data is required in order to provide labeled examples to the classifier, so that it can learn to detect the existence of a given object. A common negative dataset was used to all the objects trained. The goal of a negative set is to provide images in which there is not any of the four objects covered by this study. It was ensured that there were no faces, face silhouettes, flowers nor leaves within these images. This negative dataset was already used in [87].

Regarding positive datasets, they provide information to classifiers regarding when and where there is an object. Depending on the object classifier trained, a different dataset was used. Four datasets were constructed, one for each object. Face images were retrieved from Facity website [93], face silhouette images were fetched using a random search through image search engines, flower images were selected from Oxford 17 Category Flower dataset and other examples were added after using image search engines. Leaf images were obtained from Caltech-256 Object Category dataset and from web searches.

All the examples were binarized, in order to highlight the contrast between the object and everything around it. This decision had consequences in the image generation process because, if a classifier was trained to detect an object using only a black and white environment, it can only detect images within the same environment. Therefore, generated images are limited to black and white colours. However, image binarization also limits the search space, which is useful to reduce the difficulty of this problem, and objects within the generated images can be identified more easily by the human eye. For this purpose, a simple tool to manipulate images was developed and used to automatically binarize images. The number of examples of each dataset is presented in table 3.2.

The classifier training phase is an essential phase of the image generation process because the success of this approach depends on the generalization capabilities of the classifier.

Table 3.2: Number of examples used for each dataset

Object	Positives	Negatives
Faces	1000	1905
Face Silhouettes	55	
Flowers	312	
Leaves	201	

The executable from OpenCV had a list of parameters that could be used to tune the cascade classifier, including the number of stages. It controlled the maximum number of stages that would be trained, difficulting the task of passing all stages in order to detect an object. The minimum hit rate defined a minimum rate, per stage, of positive images that were actually classified as positive. The False Alarm (FA) rate helped to control the maximum error allowed, per stage, of negative images classified as positive. Regarding the input samples, they must respect an input size. Weak classifiers can be adjusted in terms of depth/splits (recall that weak classifiers are decision-trees). Table 3.3 summarizes the values chosen regarding these parameters.

Table 3.3: Training parameters used

Parameter	Setting
Input width	40
Input height	40
Number of stages	20
Number of splits	1
Min. Hit Rate	0.999
Max. False Alarm	0.5 (0.45 in face silhouettes)
Adaboost Algorithm	Gentle Adaboost

In order to assess the quality of the classifiers trained, they were validated using the validation dataset mentioned before. The results of this validation process are depicted in table 3.4.

Once classifiers were validated, they were integrated in the approach presented in figure 3.1, in order to allow the evaluation of images generated by the GP engine. During the execution of the evolutionary algorithm, each image was submitted to the classifier, using the object detection algorithm summarized in the steps mentioned in section 3.2. This process also required the definition of some relevant parameters, presented in table 3.5.

Concerning the GP engine, parameters chosen during these runs were similar to the ones used in previous work [87]. They are described in table 3.6.

Table 3.4: Classifiers' accuracy using a training dataset and a validation dataset.

	Training dataset		Validation dataset	
Classifiers	Positives	Negatives	Positives	Negatives
LBP Leaf Classifier	86.71%	84.24%	85.56%	92.78%
Haar Leaf Classifier	96.67%	98.01%	94.21%	98.05%
LBP Face Classifier	87.00%	98.51%	85.40%	98.44%
Haar Face Classifier	92.30%	97.52%	90.81%	95.51%
LBP Flower Classifier	98.85%	99.01%	97.19%	96.89%
Haar Flower Classifier	95.04%	99.13%	97.19%	96.89%
LBP Face Silhouette Classifier	85.45%	96.65%	-	-
Haar Face Silhouette Classifier	74.55%	93.80%	-	-

Table 3.5: Detection parameters used to evolve figurative images.

Parameter	Setting
Min. window width	42
Min. window height	42
Image Width	64
Image Height	64
Scale Factor	1.1
Image pre-processing	Otsu's Binarization

Table 3.6: GP engine's parameters used to evolve figurative images.

Parameter	Setting
Population Size	100
Generations	100
Crossover Probability	0.8
Mutation probability	0.05
Initialization Method	Ramped Half-and-Half
Initial Maximum Depth	5
Mutation max tree depth	3

3.4.2 Results

To perform this experimentation, 30 independent EC runs were done per classifier. As one studied two different classifiers per object (one with LBP features and the other with Haar features), a total of 240 EC runs were made.

The results shown presents some of the most interesting images obtained with each classifier. It is important to note that since generated images are submitted to the classifier, only images in which the classifier detected the desired object were selected. Among these set of positive generated images, a subjective assessment was made by hand, in order to select the best generated images which were considered to contain the object by the classifier. A set of 10 interesting images were gathered for each classifier, in which figure 3.6 shows faces evolved with a LBP face classifier, figure 3.7 shows faces evolved with a Haar face classifier, figure 3.8 shows face silhouettes evolved with a LBP face silhouette classifier, figure 3.9 shows face silhouettes evolved with a Haar face silhouette classifier, figure 3.10 shows flowers evolved with a LBP flower classifier, figure 3.11 shows flowers evolved with a Haar flowers classifier, figure 3.12 shows leaves evolved with a LBP leaf classifier and figure 3.13 shows leaves evolved with a Haar leaf classifier.

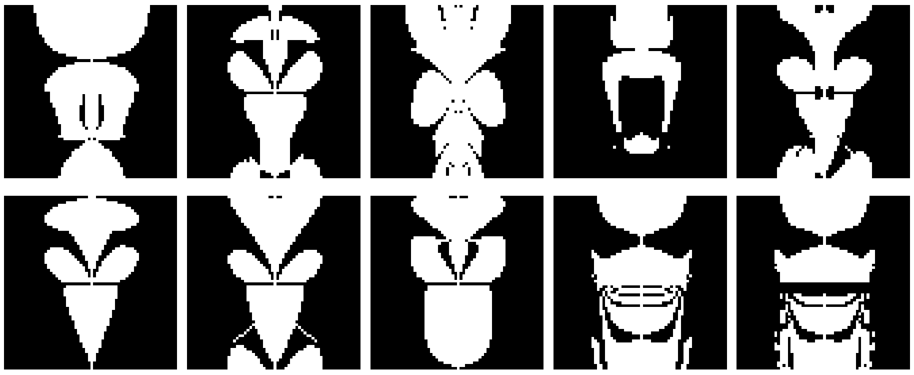


Figure 3.6: Examples of faces evolved with LBP classifier.

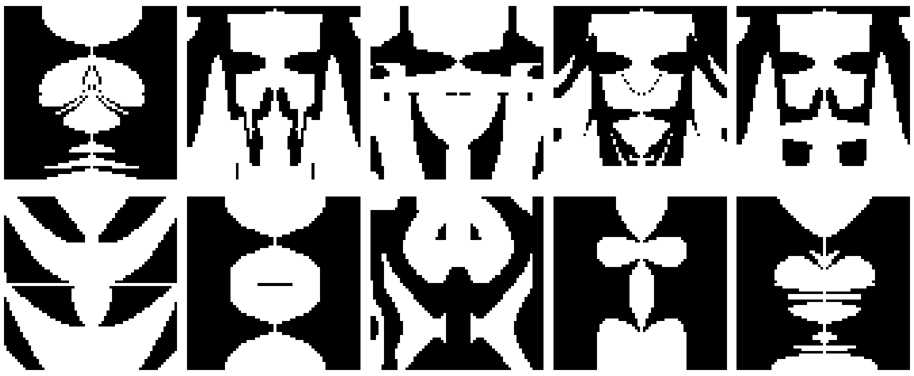


Figure 3.7: Examples of faces evolved with Haar classifier.

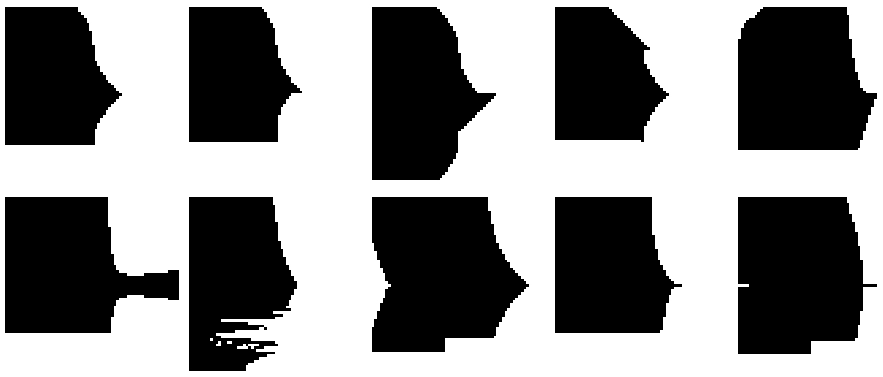


Figure 3.8: Examples of face silhouettes evolved with LBP classifier.

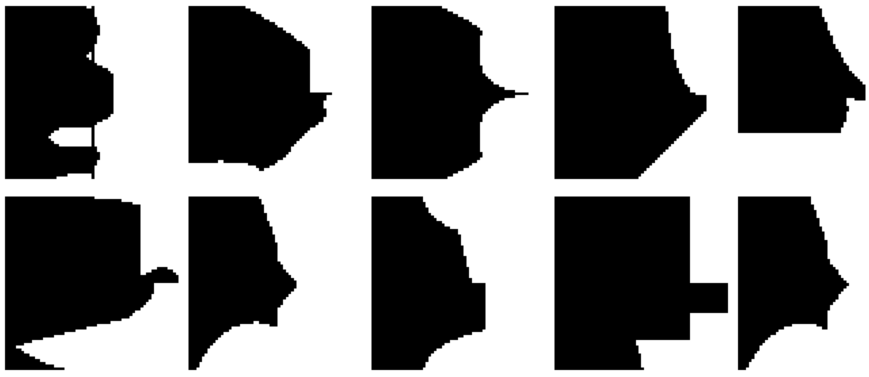


Figure 3.9: Examples of face silhouettes evolved with Haar classifier.

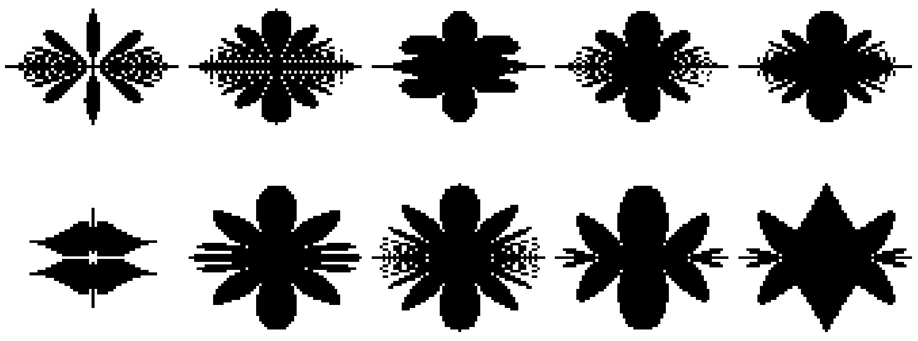


Figure 3.10: Examples of flowers evolved with LBP classifier.

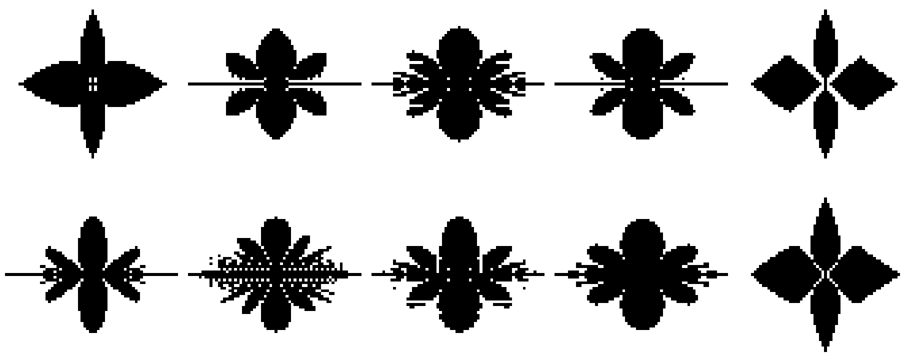


Figure 3.11: Examples of flowers evolved with Haar classifier.

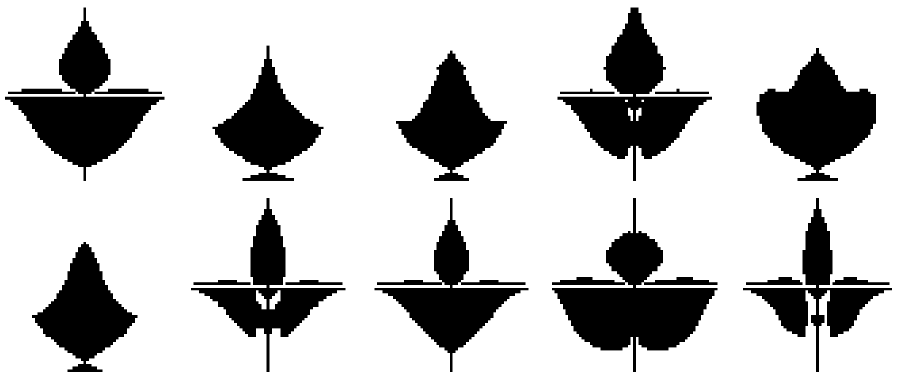


Figure 3.12: Examples of leaves evolved with LBP classifier.

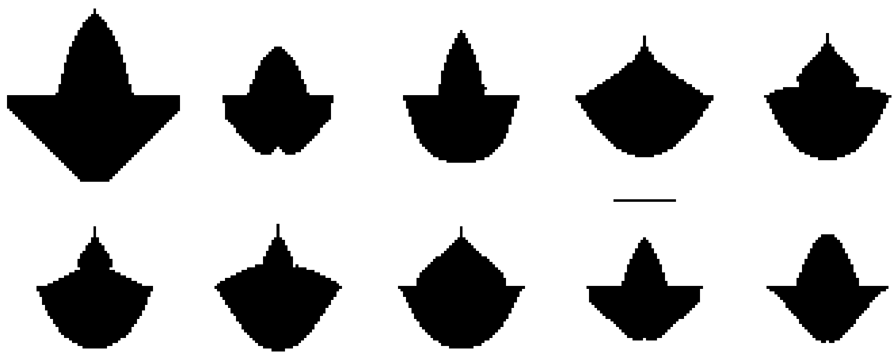


Figure 3.13: Examples of leaves evolved with Haar classifier.

These results suggest that regardless the object and the classifier chosen, the GP engine was able to generate positive images that can actually look like the desired objects. This can also be considered as a confirmation that the approach used is in fact a proposal can be applied to several objects. In this experimentation, one used objects with different degrees of complexity and distinct characteristics.

In order to prove that these images were the result of an evolution process performed during a certain number of generations and not only some images obtained fortuitously, the GP engine was analysed in terms of fitness change through generations. For each classifier, mean fitness and best fitness of each generation were stored and used them to plot graphics. Figure 3.14 shows the evolution of fitness using a LBP face classifier, figure 3.15 shows the graphic concerning a Haar face classifier, figure 3.16 shows the same situation with a LBP face silhouette classifier, figure 3.17 uses a Haar face silhouette classifier, figure 3.18 uses a LBP flower classifier, figure 3.19 uses a Haar flower classifier, figure 3.20 uses a LBP leaf classifier and figure 3.21 uses a Haar leaf classifier. Since one executed 30 runs per classifier, the metrics shown are averages of 30 runs and they are normalized according to the maximum global fitness obtained.

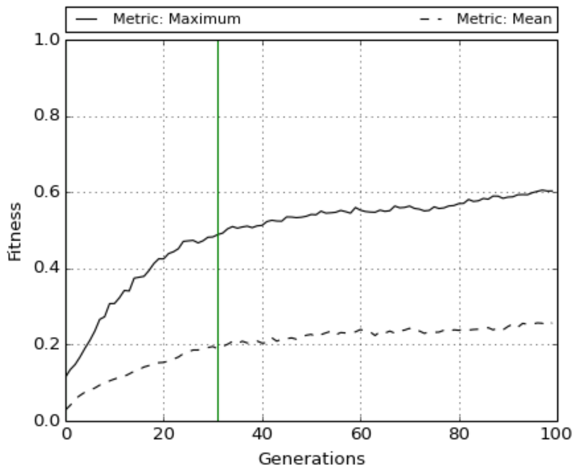


Figure 3.14: LBP faces classifier.

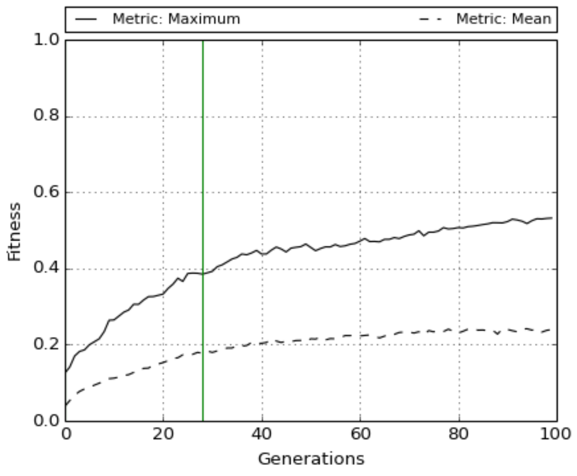


Figure 3.15: Haar faces classifier.

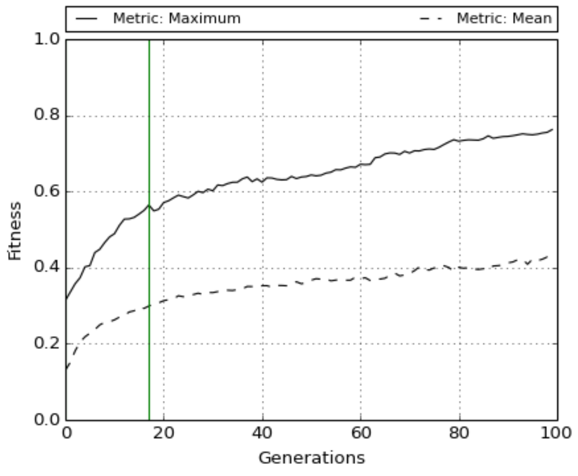


Figure 3.16: LBP face silhouettes classifier.

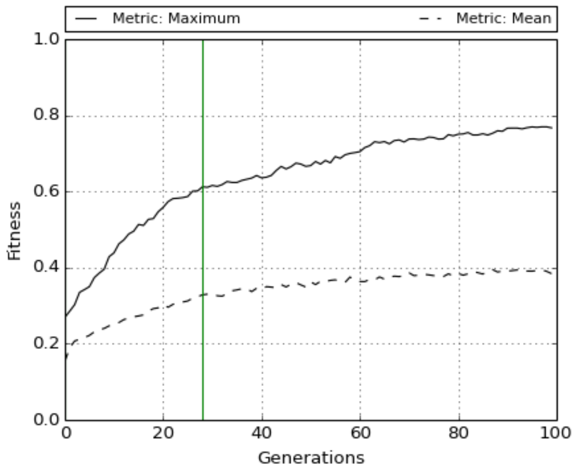


Figure 3.17: Haar face silhouettes classifier.

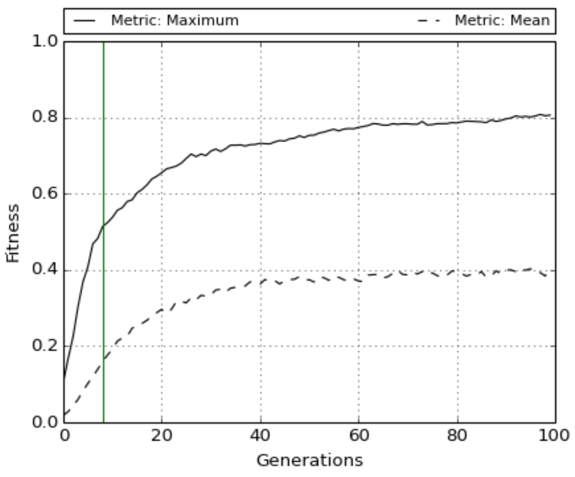


Figure 3.18: LBP flowers classifier.

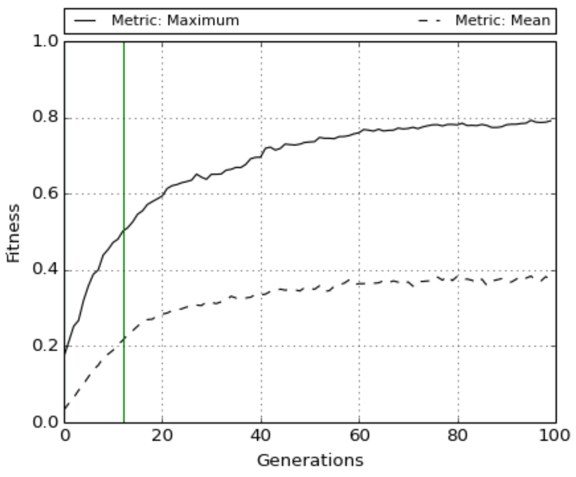


Figure 3.19: Haar flowers classifier.

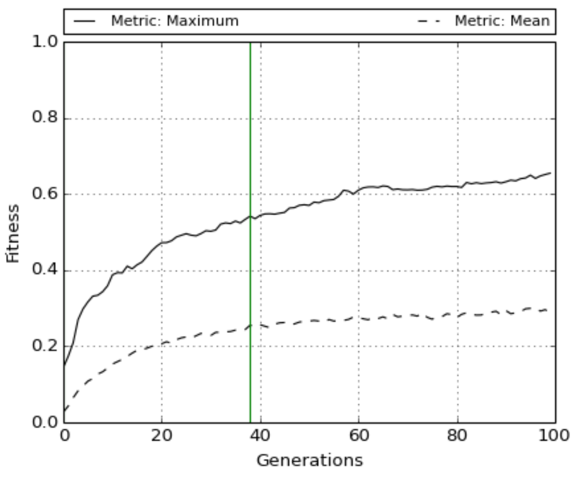


Figure 3.20: LBP leaves classifier.

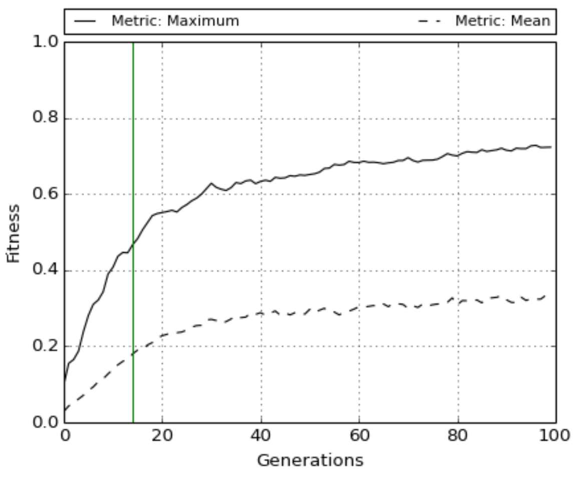


Figure 3.21: Haar leaves classifier.

These results show that the GP engine was able to evolve objects with any classifier used, in the sense that the fitness rises substantially until in the beginning of the algorithm execution and it stabilizes in the final generations. These results will be analysed in detail in section 3.4.3.

3.4.3 Analysis

From the graphics shown in figures 3.14, 3.15, 3.16, 3.17, 3.18, 3.19, 3.20 and 3.21, the first conclusion which can be drawn is that the GP engine is able to evolve figurative images that resemble the objects used to train the respective classifiers, using a few number of generations. This fact can be seen because the fitness continues to increase after the first object detection performed during the evolutionary run. In particular, figures 3.16 and 3.17, which show the fitness variation using faces silhouettes classifiers deserve a particular note, because they show that in these cases, the evolutionary algorithm converged more quickly, initializing with fitnesses substantially above when compared to the other results. This fact suggests that the problem of evolving face silhouettes is simpler than evolving any of the other three objects, which can be confirmed by observing simpler and generic shapes in the generated face silhouettes.

It is also relevant to mention that evolution towards figurative images does not imply that positive generated images will resemble objects from a human point of view. This fact stresses the importance of the classifier within the approach, in order to produce interesting results. In this case, it seems that the GP algorithm was able to somehow, exploit shortcomings of the classifier and discover unrelated images that were classified as positive. One can conclude that using these datasets, the achievement of good classification results is a necessary condition to the success of the used approach, but not a sufficient condition. Although it is beyond the scope of this work, similar results obtained in other works were a motivation to use this approach to generate images considered as false positives in order to integrate them in a new negative dataset and train new cascade classifiers with an improved generalization ability [94].

Regarding the visual analysis, figures 3.6, 3.7, 3.8, 3.9, 3.10, 3.11, 3.12 and 3.13 shows the some of the images gathered from the subjective test performed. Because it is a subjective test, it is possible to argue whether the chosen images resemble in fact an object or not.

Analysing faces figures 3.6 and 3.7, the results obtained look like robot, alien or cartoon faces. It is possible to identify the eyes in most of the cases. Mouths and noses can be identified in some faces although it is harder.

Face silhouettes are simpler objects and therefore, easier to evolve. What is possible to deduct from figures 3.8 and 3.9 is that results are more similar, the most distinctive evidence in those images is the nose, which can be smaller, bigger, or even a "pinocchio" nose.

Flowers (figures 3.10 and 3.11) and leaves (figures 3.12 and 3.13) in their turn are harder to evolve, but the GP engine seemed to have found several interesting positive generated examples which are characterized by symmetry. It is possible to find many similarities between those images, with slight changes between them. All the images have a minimalist flavour, in the sense that they only vary in terms of size and shape.

In general, images that are able to resemble the desired object from a human point of view are iconic and evolved according to the most evident characteristics detected by classifiers. In faces, they focus in the eyes, in face silhouettes they focus in the nose, in flowers they focus in petals and in leaves they tend to generate cordate ones. This evolution seems to focus only in the most important features of images and sometimes represent them in a exaggerated way, like they were cartoons.

3.5 Synthesis

The work done in this chapter can be summarized in two main tasks: the development of a tool, named norBErT, inspired on previous works, and the use of this tool to assess its ability to evolve images that resemble different objects, depending on the object classifier used.

The architecture used in norBErT involved mainly mechanisms already implemented in ECJ, concerning EC. Thus, it was necessary to construct a customized primitive set, a rendering process to generate images from trees, and integrate OpenCV API with ECJ in order to allow an external classifier to evaluate all the individuals.

These classifiers were trained with two types of features: LBP and HAAR. For each type of classifier, a training process was performed with datasets created from scratch, involving the following objects: flowers, faces, leaves and face silhouettes.

A fitness function was specifically designed taking into account the architecture of the classifier, which is a cascade of simple classifiers, each one filtering positives examples to the next stage and discarding negative ones.

The results proved that the proposed approach is able to evolve different types of objects, although it converges sometimes to search spaces where the classifier detects the object in the images generated despite they have no resemblance from an human's point of view. Furthermore, it is also noticeable that images generated from a single run are quite similar, as a result of the converging property of the GP algorithm over generations.

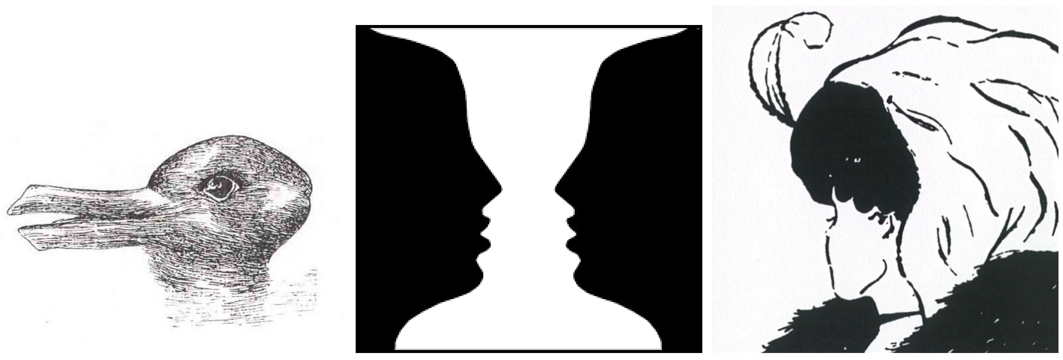
Chapter 4

Evolving Ambiguous Images

This chapter intends to study the evolution of ambiguous images through computational means using norBErT. Using several object classifiers, in a similar way when compared to the approach followed in Chapter 3, norBErT tries to attain this goal. The remainder of the chapter is organized as it follows: section 4.1 explains what ambiguity is from two different points of view, the human perspective and the computational perspective. Section 4.2 describes how the information provided by the object classifiers is used to measure the quality of the images assessed for this particular task. Then, section 4.3 contains information regarding the experimentation performed in order to evolve ambiguous images. This section is divided into section 4.3.1, which contains relevant parameters of the experimentation, section 4.3.2 that shows the results obtained and section 4.3.3 that analyses the results shown before. Finally, section 4.4 contains a synthesis of the whole chapter.

4.1 Human Ambiguity and Computational Ambiguity

Ambiguity is a well-explored concept in visual art, in which the artist builds an artwork piece that is able to be perceived in multiple and stable ways by the human brain (multistable perception). Usually, ambiguity has a negative connotation per se, as it is associated with uncertainty or doubt. However, when applied to the visual art field, ambiguity can increase the interestingness of an artwork, causing these ambiguous images to be more pleasant, despite they are harder to be analysed by the human brain [95]. Examples of some well-known ambiguous images are presented in figure 4.1.



(a) Duck/Rabbit (b) Rubin's vase (c) My Wife and My Mother-in-Law

Figure 4.1: Some well-known examples of ambiguous images.

When presenting such an artwork to a human user, he assimilates basic features from an image, such as edges or contours, and then tries to recognise something that he already knows from the set of features collected. It is relevant to mention that this recognition phase is influenced by user's memory and experience. Thus, it is possible that the image's interpretation and main deduction diverge from human to human. When an image induces the phenomenon of multistable perception in the human brain, that image is considered **human ambiguous** one.

Such a concept can be extended to the computational domain using norBErT, allowing the definition of **computational ambiguous** images. Built into the findings of Chapter 3, computational ambiguous images can be evolved with a very similar architecture, which is represented in figure 4.2.

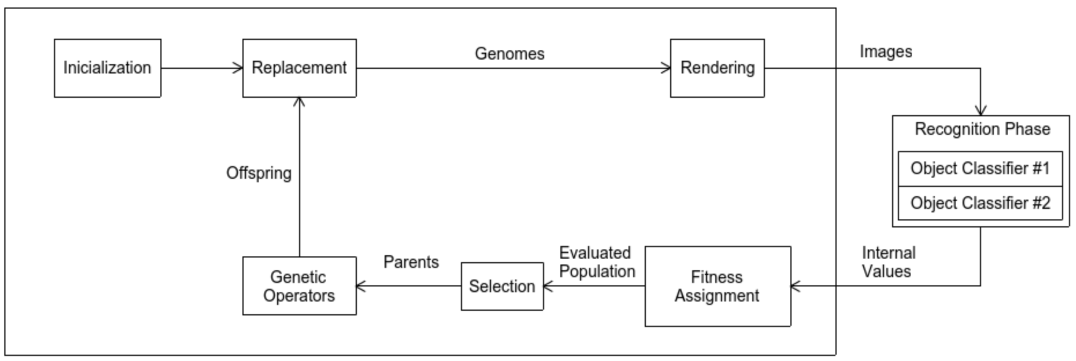


Figure 4.2: Architecture used to evolve ambiguous images.

An evolutionary algorithm is used to create these kind of images. The population is initialized, and in each generation the following process is performed: images are created from individuals' genotypes and then submitted in a recognition phase, where several classifiers assess images, providing essential information (the internal values) to assign fitness to individuals. Then, individuals are selected according to their fitness values, and mated together, creating the next generation which will be submitted to the same process all over again.

The main difference from figure 4.2 to figure 3.1 is the way images are assessed. Instead of using one classifier, as it is presented in figure 3.1, the approach used in this chapter uses two different object classifiers to assess images. Given this approach and restricting to figurative art, a **computational ambiguous** image can be defined as an image from which several objects can be perceived, within the same region. A computational ambiguous image exists when both object classifiers are able to detect their respective object within the same image region.

4.2 Aggregated Fitness Functions

Given that one is coping with several classifiers to help in the fitness assignment task, and those classifiers provide a binary output, it is important not only to gather internal information from them in order to have a suitable fitness landscape but also to merge quality measures from both object classifiers. This step is needed in order to determine which individuals will survive.

Recall from chapter 3 that these classifiers have a cascade architecture, where each step of the cascade is a different stage. Images are analysed in each stage, where they can pass to next stages if they deserve further analysis and eventually get the end of the cascade if an object was detected, or they can be discarded.

As this new task has increased complexity, fitness assignment of potentially ambiguous images is divided into two phases.

The first phase consists in measuring a degree of resemblance of the image with each object desired, using the respective object classifier. This process is very similar to the fitness assignment scheme explained in 3.2.

$$f(x) = \sum_{i}^{cstages_x} (beststagedifference_x(i) * i) + cstages_x \cdot 100 + detected \cdot 2000 \quad (4.1)$$

Variables $cstages_x$ and $beststagedifference_x(i)$ are extracted from the classification intermediate information. The rationale is that an image that passes several stages has a higher *cstages* value and is likely to be closer to being recognized as having a object than one that passes fewer stages. Images that are clearly above the thresholds associated with each stage have higher $beststagedifference$ values. As such, these images are preferred over ones that are only slightly above the threshold. Additionally, the *detected* variable is set to 1 if it was detected an object within the image, otherwise it is set to 0. This allows norBErT to favour the selection of images which actually contain an object.

The second phase can be interpreted as a multiobjective optimization problem resolution, where each objective is the resemblance to an object, using a plain aggregation technique. As one is interested in evolving ambiguous images where two objects can be equally identified, equal weights are associated to both objectives. The final fitness function is described in equation 4.2

$$combined(x) = \prod \log_2(f_i(x) + 2), \tag{4.2}$$

where $f_i(x)$ is the ith single fitness function (4.1), per classifier of the combination.

4.3 Experimentation

This study covers the evolution of leaves and faces simultaneously, in a trial to evolve ambiguous images. A similar study was performed using faces and flowers, and despite it is not covered in this chapter, there was some work done using these two objects, which can be observed in appendix E.

The experimentation performed in this chapter shares more similarities with the one performed in chapter 3. In a preliminar phase, new classifiers were trained, in slightly different conditions, in order to assess the influence of classifiers' robustness in the results. However, in this chapter one uses the same datasets gathered in chapter 3 to train classifiers. The type of classifiers adopted to this study was merely LBP, since it was prooved in chapter 3 that the evolution of interesting images was possible with both Haar and LBP classifiers and the latter ones are quicker in the training process.

In a second phase these leaf and face classifiers are applied in norBErT's approach, as described in figure 4.2, in order to assess the capability of evolving ambiguous images. Section 4.3.1 describes the parameters chosen in each step, section 4.3.2 presents relevant results concerning this study and section 4.3.3 interprets and analyses them.

4.3.1 Setup

The setup planned for this study can be quite complex to understand given that there are a considerable number of parameters regarding distinct parts needed to enable the image evolution task. For this reason, the setup is divided by each one of these parts.

Classifier creation

In this approach, the classifiers used play the role of supervisor by helping in the fitness assignment task. In order to create these classifiers it is necessary to train and validate them.

The process followed to create new classifiers has many things in common when compared to chapter 3. In this study one intends to obtain classifiers which are more robust, using the same datasets to train them. For further information about the datasets, check section 3.4.1.

The cascade classifiers, originally proposed by Viola and Jones [6], were again the choice as the classifier to integrate in norBErT's approach. The training process was the same, using the same tool provided by OpenCV to create the classifier. The parameter set used to train these classifiers is described in Table 4.1. Further explanation about each parameter can be found in section 3.4.1, or alternatively, in OpenCV documentation[1] or in [6].

Table 4.1: Training parameters used to evolve ambiguous images.

Parameter	Setting
Input width	40
Input height	40
Number of stages	30
Number of splits	1
Min. Hit Rate	0.999
Max. False Alarm	0.5
Adaboost Algorithm	Gentle Adaboost

Comparing to table 3.3, all the parameters remain the same but one, the number of stages. The rationale behind the increase of the number of stages (changed from 20 to 30) is the following. Considering that in each stage an image can be discarded or it can advance to the next stage if it deserves further analysis, increasing the number of stages will increase the probability of discarding the image before it gets the end

[1]`http://docs.opencv.org/doc/user_guide/ug_traincascade.html`

of the cascade. Therefore, the risk of having a negative sample classified as positive (contains an object) is lower (recall that the main concern identified in previous works is the high false positive rate).

As in the experimentation performed in chapter 3, the classifiers were validated within the same conditions. The result of this process is presented in Table 4.2. Note that an additional flower classifier was trained and validated. Albeit it is not used in this study, it is useful in other experimentations.

Table 4.2: Classifiers' accuracy using a training dataset and a validation dataset

	Training dataset		Validation dataset	
Classifiers	Positives	Negatives	Positives	Negatives
LBP Leaf Classifier	77.46%		69.44%	99.80%
LBP Face Classifier	85.70%	99.13%	85.23%	99.80%
LBP Flower Classifier	70.61%	99.62%	69.15%	99.61%

Object Detection Algorithm

As it is presented in figure 4.2, the evolution of ambiguous images implies the existence of classifiers that are able to detect the presence (or absence) of an object within a given image. For further information about the object detection algorithm, check section 3.2. The relevant parameters regarding the object detection algorithm are presented in Table 4.3.

Table 4.3: Detection parameters used to evolve ambiguous images.

Parameter	Setting
Min. window width	90
Min. window height	90
Image Width	128
Image Height	128
Scale Factor	1.1
Image pre-processing	Otsu's Binarization

Comparing to the experimentation performed in chapter 3, there are noticeable changes in some parameters values.

The size of the images generated is increased from 64x64 to 128x128. This allows the creation of larger objects, favouring the detail of the images generated. Besides, the minimum window size used in this experimentation is also increased, from 42x42 to 90x90. This enforces different objects to be located in overlapped regions within the image. It is also relevant to mention that images are transformed before they are analysed, using the Otsu's binarization algorithm, which causes images to be analysed and evolved with only black and white colours. The main reason behind this decision is that binarized images tend to have an easier an clearer interpretation to humans.

Evolutionary Algorithm

In order to run the evolutionary algorithm that will be responsible to create the ambiguous images autonomously, some parameters need to be set as well. Following the reasoning of the experimentation in chapter 3, the set of parameters chosen is presented in Table 4.4.

Table 4.4: GP engine's parameters used.

Parameter	Setting
Population Size	100
Generations	1000
Crossover Probability	0.8
Mutation probability	0.05
Initialization Method	Ramped Half-and-Half
Initial Maximum Depth	5
Mutation max tree depth	3
Elite size	1

Regarding the evolutionary algorithm, the number of generations was increased from 100 to 1000 when compared to the experimentation of chapter 3. Also, the best solution of each generation is preserved and automatically passed to the next generation. These two changes are justified by the increasing difficulty of evolving images which contain several several objects instead of just one. As an aggravating, these objects should appear in overlapped regions of the image.

4.3.2 Results

In this experimentation two different situations are tested: the use of permissive classifiers (created during the experimentation in chapter 3) and the use of robust classifiers, specified in table 4.2. For each situation, 30 runs are performed, using two classifiers at the same time to guide evolution: a leaf classifier and a face classifier. Figure 4.3 depicts the fitness of the best individuals (average of 30 runs). The fitness that is taken into account for evolution purposes is the combination of the fitnesses from the two classifiers. Additionally, each partial fitness (regarding each object) is shown. Dashed lines correspond to the situation where permissive classifiers are used whereas solid lines correspond to the evolution of objects using robust classifiers. Each fitness value is normalized according to each maximum value obtained with the respective classifier.

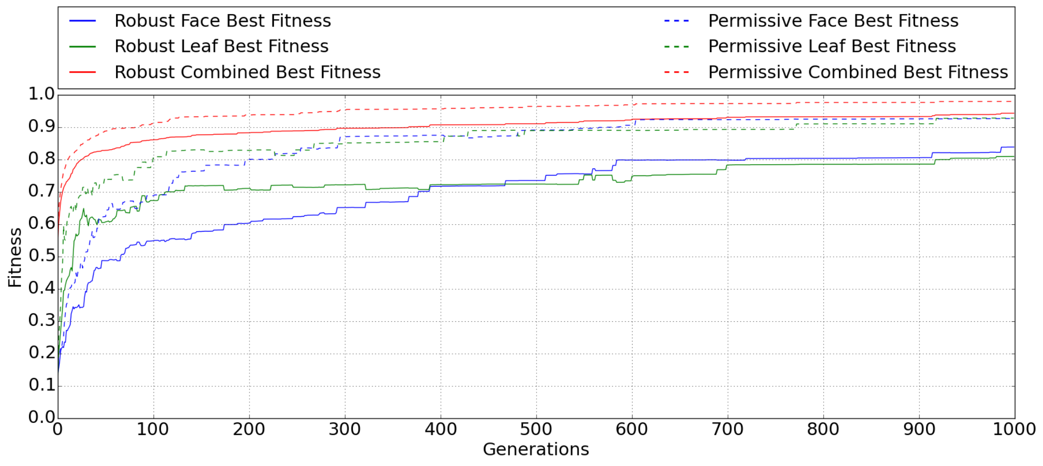


Figure 4.3: Evolution of the fitness of the best individual along the evolutionary run (results are averages of 30 runs).

Figure 4.4 shows the average detection rate of the best individuals in each generation. It contains the percentage of best individuals which contains an object detected by the classifier.

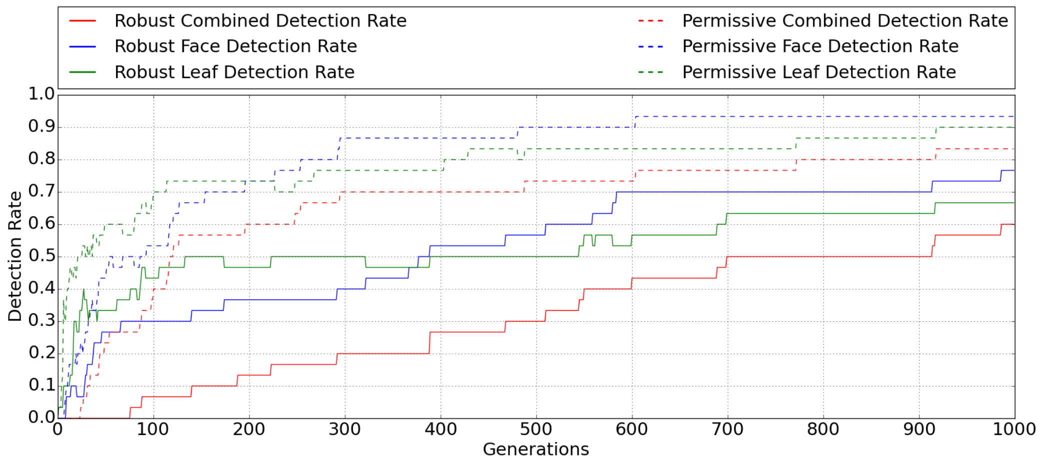


Figure 4.4: Best individual's detection rate along the generations.

These results show that the fitness function favours the appearance of images which contain simultaneously leaves and faces. This conclusion was obtained due to the similar characteristics of the red line in both figures 4.3 and 4.4. Both tend to increase and converge towards a given value.

Considering a successful run as a run that is able to evolve as least one image where a leaf and a face detected on that image, the robust situation was able to produce 60% of successful runs whereas the permissive case was able to produce 83.3% of successful runs. Continuing the comparison between the permissive and robust experimentations, and using the 100th generation as a reference, the robust situation produced less than 10% of successful individuals, whereas the permissive situation produced nearly 35% of successful individuals.

Comparing the different components of the fitness functions, for both situations, two main results can be extracted. From figure 4.3, the fitness values from the individual components are lower than the combined fitness value. Regarding figure 4.4, the number of detections over the best individuals considering a single object (face **or** leaf) is higher than the number of detections considering the combination of both components.

4.3.3 Analysis

From the results observed in section 4.3.2, the first conclusion drawn by the results obtained is that the problem of evolving images containing two overlapping objects is harder than evolving images which resemble a single object. Not only the final combined detection rate (red line from figure 4.4) is lower than the single components' detection rate (blue and green lines from figure 4.4), but also it sounds logical that a multiobjective optimization problem is harder to solve than a simple optimization problem.

There are two factors identified that can influence the added difficulty of evolving ambiguous images. First, the classifiers' characteristics can influence the success of this approach. From figures 4.3 and 4.4, it is possible to observe that more permissive classifiers ease the resolution of this problem (all the dashed lines are above the correspondent solid lines in terms of fitness values and detection rate). However, permissive classifiers are less accurate which makes the probability of generating images that do not resemble the desired object(s) from a human perspective higher. The other aspect is the pair of objects chosen to evolve together. The difficulty of evolving ambiguous images depend also on the graphical compatibilities evidenced by the two images. Considering this problem as a multiobjective evolution problem, one can deduct that the higher are the single fitness components of this problem, the easier is the task of evolving them assembled, because the set of objectives are not so conflicting between them. In this particular case, for the permissive case, the single components (leaves and faces) have higher detection rates values when compared to a side-study which is depicted in appendix E and uses faces and flowers.

Another interesting aspect to observe is that it seems easier to evolve leaves than faces in the beginning. Observing figure 4.3, in the first 100 generations fitness values from the leaf component achieve higher values when compared to the face component. Then, the aggregated fitness function favours the quality improvement of the weakest component, and around the 400th generation both components achieve similar values. From that moment on, improving the combined component is an harder task.

Analysing the images obtained, although a substantial number of runs produced images which are computationally ambiguous, they fail to induce the same ambiguity from a human perspective. Examples of these images are presented in figure 4.5.

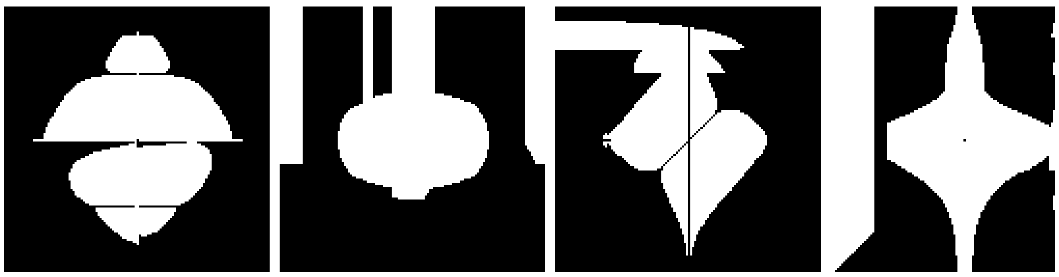


Figure 4.5: Images that are ambiguous from the computational perspective, but were not considered as ambiguous from a human perspective.

However, there were cases where it was possible to detect ambiguity from both human and computational perspective. Examples presented in figure 4.6 depicts these kinds of images. It should be noted that the choice of these images is arguable and subjective and their human ambiguity identified from the author's point of view.

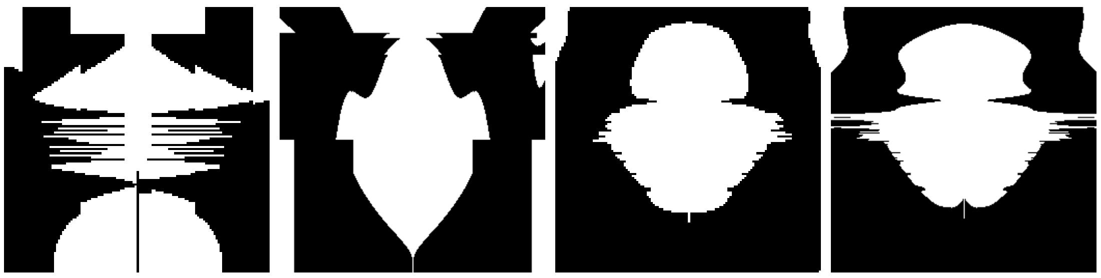


Figure 4.6: Images considered as ambiguous from both computational and human perspective.

The images presented in figure 4.6 have different shapes and contours. However, the ambiguity effect produced is similar in all the cases. The face object seems easier to detect because, as a human, we are trained to recognize faces. All the faces look like masks where a white-shaped are has a face contour and the black areas which penetrate in the white part look like eyes or eyebrows. The leaf object can be detected by looking only in the white area under the eyes zone. Besides, the leaf stalk is present in all the images' bottom region.

4.4 Synthesis

The work performed in this chapter consisted in extending the approach explored in chapter 3 to evolve images towards several objects. This is done by using two object classifiers, instead of one, to classify images and detect the presence of both objects. The final objective of this work was to force both objects to appear in overlapping regions of the produced images and analyse whether they could be detected as ambiguous to the human eye.

As each classifier produced an independent fitness component, the solution found to evolve both objects was to merge each component's information using a combining fitness function.

The experimentation done involved a pair of robust and a pair of permissive classifiers. The study performed involved the evolution of faces and leaves simultaneously. The results obtained show that this problem is harder to solve than the problem depicted in chapter 3, and its hardness depends on the classifiers' robustness. Therefore, there is a drawback between the images' ability to resemble the desired objects and the approach's ability to solve the problem successfully. The approach is able to solve the problem and create some interesting where both objects can be detected by the human eye and they can even be considered ambiguous images from a human perspective. However, in most of the cases, images can hardly be considered as ambiguous from a human point of view, in the sense that they do not resemble the desired objects.

Chapter 5

Evolving Images through Novelty Search

This chapter intends to attain one of the main goals of this work, evolve a wider range of objects, building a set of object images which are visually different from each other.

Based on the findings of chapter 3, the problem of getting similar object images is tackled by promoting an evolution mechanism which rewards the uniqueness of an individual when compared to others, inspired by novelty search.

The remainder of this chapter is organized as it follows. Section 5.1 highlights the changes performed on the original approach, in order to be able to enable evolution according to several criteria. Section 5.2 provides an insight into how novelty is computed, so that the fitness of each individual could be measured in terms of uniqueness. Then, section 5.3 explains how fitness and novelty are combined in order to promote a multiobjective approach, called hybrid evolution in the remainder of this chapter. The exploration and study of the various mechanisms were applied to the evolution of figurative images, in section 5.4, and the evolution of ambiguous images, in section 5.5. Finally, an overview of this chapter is presented in section 5.6.

5.1 An evolutionary engine with uniqueness reward

In section 3.1, a general purpose approach was proposed in order to evolve figurative artworks. This evolution was performed with a fitness-based approach, using internal values from an object classifier to help in the fitness assignment task. In this section, a modified version of the architecture used in section 3.1 is suggested. These modifications allow the evolution of figurative images taking into account both fitness and their uniqueness when compared to other images. This new architecture is shown in figure 5.1. Note that only the components marked in red are new or suffered modifications for novelty purposes.

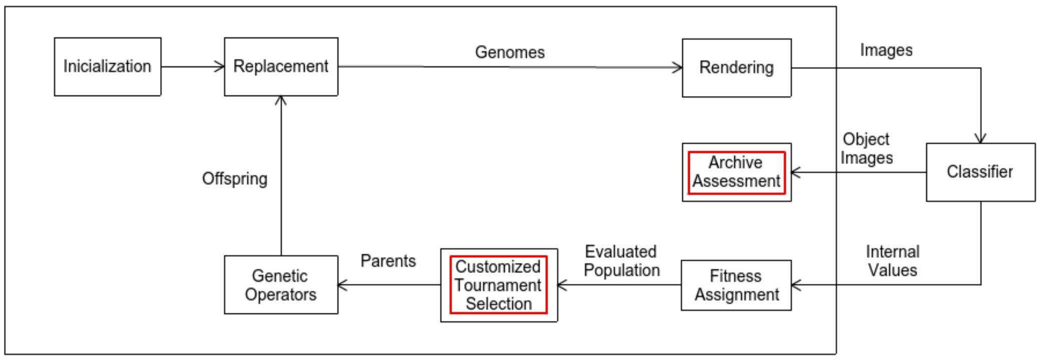


Figure 5.1: Architecture with novelty reward used for the experimentation work.

Despite the architecture described in figure 5.1 including components which enable evolution guided by novelty, the old fitness-based evolution was not discarded. This decision was made for two reasons. First, because it made possible to evolve images according to three different modes: a fitness-based one, which was already explored in chapter 3, a novelty-based evolution, which evolves images according to their uniqueness, and a third mode called hybrid mode, which considers novelty and fitness as two different objectives and performs evolution trying to maximize both. The second reason, which also explains the existence of a hybrid evolution mode, is that novelty search, when applied *per se*, is not very effective in problems with large search spaces, as it is mentioned in the literature [96].

Briefly explaining, in the architecture described in figure 5.1, a given number of individuals are randomly created for the initial population. Then, an iterative process is started with the transformation of the genomes into images (the rendering process). Images are then evaluated externally by an object classifier, which is responsible to assess whether the candidate image contains the object or not, and more importantly, it provides internal values which help to the determinine resemblance degree of the candidate image to a given object in the fitness assignment phase. Moreover, the object classifier has a new role within the architecture described in figure 5.1, by filtering the images in which an object is detected to an archive. Before being admitted into this archive, images are assessed in order to verify if they are novel enough to be part of the archive. In short words, each new image is compared to the ones that already exist in the archive. If its uniqueness degree remains above a threshold, the image is added to the archive, otherwise, it is discarded. Detailed information about novelty archive assessment is provided in section 5.2. The archive has two functions, it helps evolving images in terms of novelty and it can be used to analyse norBErT's ability to evolve different images. The remainder of the process is followed by a tournament selection, which selects the parents that will be used to generate the offspring by applying genetic operators, and finally, the children individuals will be used to fill the new population, by replacing the old one.

In terms of novelty, the tournament selection deserves special focus, because novelty is computed in this phase, depending on the individuals selected to be part of the tournament. Besides, each tournament winner can be selected in this phase according to three different criteria: best individual in terms of fitness, best individual in terms of novelty, and best individual tanking into account both fitness and novelty (hybrid selection). More information about this novelty assignment is described in section 5.2.

5.2 Computing novelty in norBErT

Observing figure 5.1, there are two steps which involve novelty mechanisms, the archive assessment and the customized tournament selection. To provide better understanding about norBErT's working mode with novelty, two different concepts will introduced: **archive novelty** and **evolving novelty**. The first one is used to verify whether an image should be added to the archive (archive assessment phase), while the other one is used to determine the uniqueness degree of an individual, for selection purposes.

Archive Novelty

An archive is used to evaluate norBErT's capability of generating content in terms of quantity and diversity. The point of archive novelty is to assess whether an evaluated image meets the requirements needed to be a unique solution. Bearing this idea in mind, any image where an object is detected by the object classifier is evaluated in terms of archive novelty. One can denominate any image in this situation a **candidate image**.

Archive novelty computation relies on distance comparisons, using dissimilarity measures, between a candidate image and a set of images already in the archive. If a candidate image is found by the classifier and there are no images in the archive yet, the candidate image is automatically added. If there are any images in the archive, archive novelty *archnov* for a given candidate image i is calculated according to equation 5.1

$$archnov(i) = \frac{\sum_{j=1}^{n} d(i,j)}{n} \tag{5.1a}$$

$$n = \begin{cases} sizearchive & \text{if } sizearchive < max \\ max & \text{if } sizearchive \geq max \end{cases} \tag{5.1b}$$

where $d(i,j)$ is a dissimilarity measure between the candidate image i and the jth image in the archive. In order to compute *archnov*, the n most similar comparisons between the candidate image and the images in the archive are used. Consequently, the computation of archive novelty implies a sorting between all the pairwise distances between a candidate image and the images in the archive. The number of n comparisons also varies depending on the size of the archive and it is calculated according to equation 5.1b. This number will be equal to the archive size, denoted by *sizearchive*, unless it reaches a predefined parameter value, denoted by *max*. In that case $n = max$. After computing the archive novelty of a candidate image, if its value passes a given threshold, the image is added to the archive. The entire process of novelty archive assessment is described in algorithm 2.

Evolving Novelty

The concept of evolving novelty is computed during the selection phase, at the tournament level. As this process may be harder to understand, the explanation of this novelty type is based on an example. Consider a population of 10 individuals, a tournament size of 5, and 4 individuals in the novelty archive. For a tournament, 5 individuals are selected randomly from the population, in order to determine which one is the winner. Then, for each individual picked *ind*, comparisons using a dissimilarity metric are made between *ind* and the other individuals picked for the tournament. The same measure is retrieved using *ind* and the images from the novelty archive. All these distances are sorted and the nearest 4 are used to compute the evolving novelty of *ind* ($evolnov(ind)$) according to equation 5.2.

Regarding the evolving novelty computation, it requires that individuals must be rendered again. As the dissimilarity metric processing time depends on the number of pairs compared of images being compared and the size of the images, one chose

```
Procedure: NoveltyArchiveAssessmentAlgorithm
input:
    – A candidate image (candidate);
    – An archive (archive) containing previously added entries;
    – A threshold (threshold) to decide if candidate is added to archive;
    – A max parameter to look up for the max most similar images;
begin
    distances ← $\emptyset$;
    archnov ← 0;
    foreach image $j \in$ archive do
        dissimilarity ← dissimilaritymetric(candidate, j);
        add dissimilarity to distances;
    end
    distances ← sort(distances) ;                    // in ascending order
    if length(archive) $\geq$ max then
        n ← max;
    else
        n ← length(archive);
    end
    for $j \leftarrow 1$ to n do
        archnov ← archnov + distances[j];
    end
    if archnov $\geq$ threshold then
        add candidate to archive;
    end
end
```

Algorithm 2: Archive Novelty Assessment Algorithm.

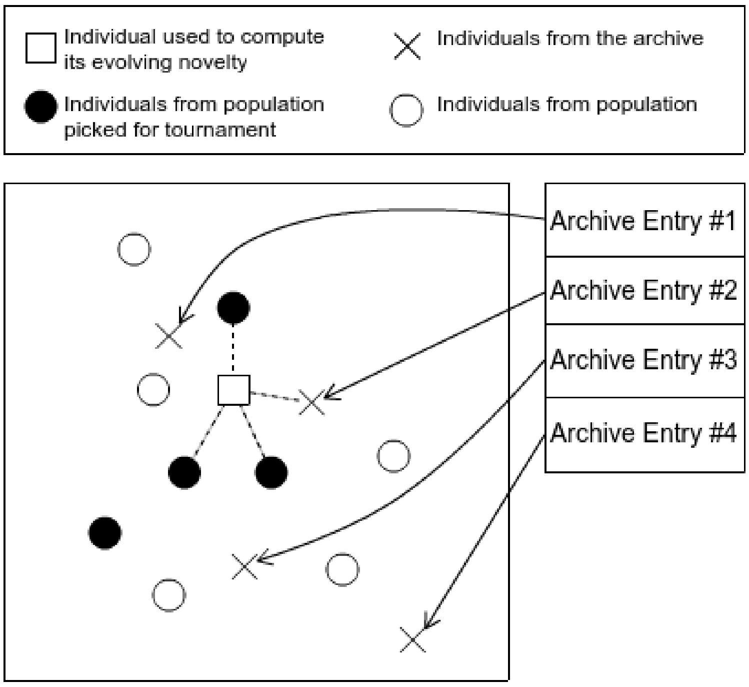
to render images in the tournament selection phase with a lower resolution (20x20). This allows the algorithm to run faster, without losing too much detail.

$$evolnov(i) = \sum_{j=1}^{tournsize-1} d(i,j) \tag{5.2}$$

To provide a better understanding of the whole process, algorithm 3 describes how evolving novelty is computed for each individual, and figure 5.2 illustrates the same process for this example. In the latter, it is also showed which distances were chosen to compute evolving novelty using dashed lines.

```
Procedure: EvolvingNoveltyComputationAlgorithm
input :
    – An individual whose respective image will be calculated in terms of evolving
      novelty (ind);
    – The archive containing all the distinctive images found so far (archive);
    – The set of individuals picked for the tournament (tournament);
output: The evolving novelty evolnov of individual ind

begin
    distances ← $\emptyset$;
    evolnov ← 0;
    remove ind from tournament;
    foreach individual $j \in$ tournament do
        dissimilarity ← dissimilaritymetric(ind, j);
        add dissimilarity to distances;
    end
    foreach image $j \in$ archive do
        dissimilarity ← dissimilaritymetric(ind, j);
        add dissimilarity to distances;
    end
    distances ← sort(distances) ;                    // in ascending order
    for j ← 1 to length(tournament) do
        evolnov ← evolnov + distances[j];
    end
    return evolnov;
end
```
Algorithm 3: Evolving Novelty Computation Algorithm.



Individual used to compute its evolving novelty
Individuals from the archive
Individuals from population picked for tournament
Individuals from population
Archive Entry #1
Archive Entry #2
Archive Entry #3
Archive Entry #4

Figure 5.2: Schematic representation of how evolving novelty is computed.

Some decisions regarding evolving novelty should be highlighted and justified, because they are slightly different when compared to the original novelty search proposal [45].

The archive is an utility both used in this proposed approach and in the original novelty search approach. Both are useful to help in (evolving) novelty computation and precluding the algorithm from performing a search similar to "random search", by ensuring that solution in the archive are not visited repeatedly. However, the archive from the original novelty search proposal saves any solution whose (archive) novelty value (which takes into account the population and the archive) attains a given threshold, whereas this approach is used to store only feasible solutions (containing an object) which are novel enough when compared only to that archive. Within these conditions, this work's archive can help in the (evolving) novelty calculation task, while being an useful performance evaluation tool, as it will contain feasible and novel solutions, which allows to assess the quantity and diversity of the produced images. Besides, the number of entries in the archive will be lower, which consequently saves time and computational burden.

The exploitation of tournament selection in order to compute (evolving) novelty is a distinctive procedure as well. While in the original version, (evolving) novelty is computed picking the n nearest distances taking into account the whole population and the archive this approach uses only individuals from the tournament and the archive. This decision was made due to time restrictions in terms of algorithm execution, allowing to save time and computational burden.

5.3 Combining fitness and novelty

Among the literature about novelty search, there is some reluctance in applying only novelty search to solve any problem [61]. These works argue that the use of only novelty search in problems with big search spaces would not be a good way to tackle the problem because it could take a long time until it gets satisfactory solutions. In fact, in these situations, novelty search characteristics induce that finding an interesting solution without a goal can be difficult as the solutions have to disperse throughout the search space to find any interesting solution. Furthermore, if an almost interesting solution is found, novelty search is not aware of that and it will find solutions based on individuals' novelty only. Within these conditions, there has been a tendency to choose an evolution which combines both fitness and novelty.

In this work, novelty and fitness are treated as two different objectives. They are conciliated in an evolutionary algorithm through a multi objective evolutionary algorithm, which uses a Pareto-based approach at the tournament level. In essence, all the dominated individuals are excluded from the set of possible winners, and in this final set of possibilities one has all the non-dominated individuals. From this set, the winner is chosen randomly. Algorithm 4 describes the process of choosing a winner in each tournament using this approach. For more information, concepts related with Pareto-based approaches are explained in section 2.1.3.

5.4 Experimentation - Evolving figurative images

This experimentation covers the search for a proper mechanism which uses novelty search and its application on the evolution of figurative objects. In order to perform this study, one used object classifiers concerning flowers, faces and leaves (the ones trained and validated as it is presented in table 4.2).

As in previous chapters, section 5.4.1 presents the parameters used in this experimentation, giving more relevance to the ones who were created or modified in this phase.

```
Procedure: HybridTournamentSelection
input  : A set containing the eligible individuals of being selected
         (tournament)
output: The winner (winner) of the tournament (tournament)
begin
    for $i \leftarrow 1$ to length(tournament) do
        for $j \leftarrow i + 1$ to length(tournament) do
            if $tournament[i] \succ tournament[j]$ then
                remove $tournament[j]$ from tournament;
            else if $tournament[j] \succ tournament[i]$ then
                remove $tournament[i]$ from tournament;
        end
    end
    winner $\leftarrow$ random(tournament);
    return winner;
end
```

Algorithm 4: An hybrid tournament selection which takes into account novelty and fitness

Due to novelty search behaviour in large search spaces, there was a need to perform a deeper study of its application in this particular problem. Only then, a direct comparison between fitness and novelty was made, in order to assess the phenotypic range and diversity provided by each mechanism. Section 5.4.2 depicts these results in a clear and neat way. Then, these results are analysed in section 5.4.3.

5.4.1 Setup

As described in previous studies, the setup phase inolves several parameters regarding different parts of the whole approach. For this reason, the parameters presented are divided in four main parts.

Classifiers

In opposition to previous experimentations, there were no new classifiers created. Instead, the ones trained and presented in table 4.2 were used. In a first phase, during the search for a proper novel mechanism, only a flower classifier was used. Then, the experimentation was extended to faces and leaves as well.

Classifiers are important to help in the fitness assignment task. Regarding the fitness function, the one in section 4.2 (namely, equation 4.1) was adopted.

Object Detection Algorithm

The parameters concerning the object detection algorithm are the same as the ones used in the experimentation in 3. For further information regarding these parameters, observe table 3.5.

Evolutionary Algorithm

In order to be able to generate images, norBErT parameters need to be set up regarding its evolutionary algorithm. For this experimentation in particular, there are a few parameters added or changed. Nevertheless, most of them were already used in the experimentation performed in chapter 4. The parameters of this experiment are presented in table 5.1.

Table 5.1: GP engine's parameters used

Parameter	Setting
Population Size	100
Generations	500
Crossover Probability	0.8
Mutation probability	0.05
Initialization Method	Ramped Half-and-Half
Initial Maximum Depth	5
Mutation max tree depth	3
Elite size	1
Tournament Size	5
max parameter	5
Dissimilarity Metric	Root Mean Squared Error
Archive Threshold	100

The new parameters, namely the tournament size, parameter *max*, the dissimilarity metric and the archive threshold were added to this table due to the new parameters demanded by novelty techniques. Recall that the *max* parameter denotes the the maximum number of individuals from the archive taken into account for archive novelty computation (described in detail in 5.2) and the archive threshold is equal to 100. Given that an image is rendered with the resolution 20x20 for novelty purposes, this means that an image needs to have 20% of different pixels to enter in the archive.

Tests Setup

In order to find the ideal way to evolve images with the aid of novelty search, different types of evolution were tested in norBErT. These range of tests can be be summarized to five different strategies.

Fitness only evolution This is the strategy already used in chapters 3 and 4. The evolution is performed according to the resemblace of each individual to a given object.

Novelty only evolution The evolution is performed according to an uniqueness degree which is calculated at the tournament level with a pairwise dissimilarity metric. For further details, consult section 5.2.

Hybrid evolution This type of evolution consists in considering fitness and novelty as two different objectives at the tournament level. For further information about this multiobjective approach, see section 5.3.

Two-phases evolution Approach 1 This approach consists in applying a fitness evolution in a first phase, until an object is detected by the object classifier. From that moment on, the hybrid evolution is applied to norBErT.

Two-phases evolution Approach 2 This approach consists in switching between fitness and hybrid evolution according to some criterion. For the instance, this approach starts with fitness evolution until 15% of the population is detected as containing an object. If it passes this threshold, there is a switch to the hybrid evolution. This hybrid evolution is maintained until the percentage of the population which contains the desired object drops below 5%. In this case, norBErT changes back to fitness evolution.

5.4.2 Results

This experimentation is divided in two parts. The first one consists in running norBErT using only a flower classifier, but with the different evolution techniques described in the setup. Figures 5.3, 5.4 and 5.5 depict the results obtained in the first part as an average of 30 runs. Figure 5.3 presents a chart of how the fitness of the best individual varies along the generations. As for figure 5.4, it shows in the same 30 runs how the detection rate of the best individual varies along the generations.

Fitness results are normalized by dividing their raw results by the maximum value achieved.

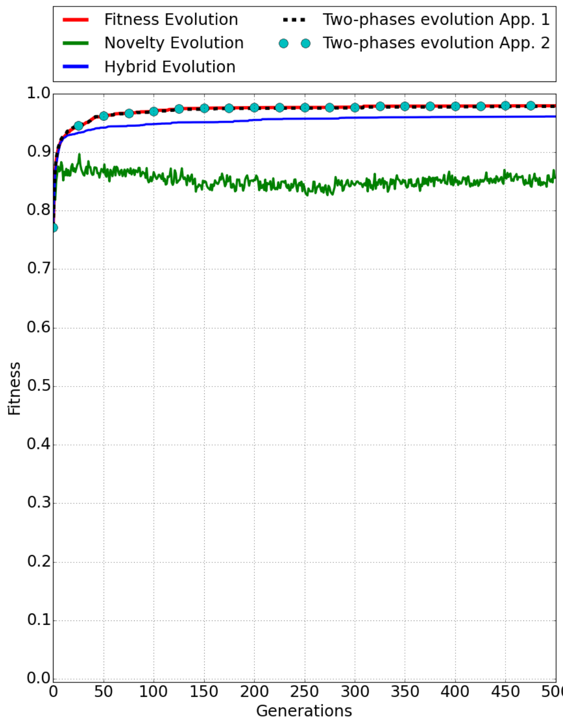


Figure 5.3: Evolution of the fitness of the best individual along the evolutionary run (results are averages of 60 runs).

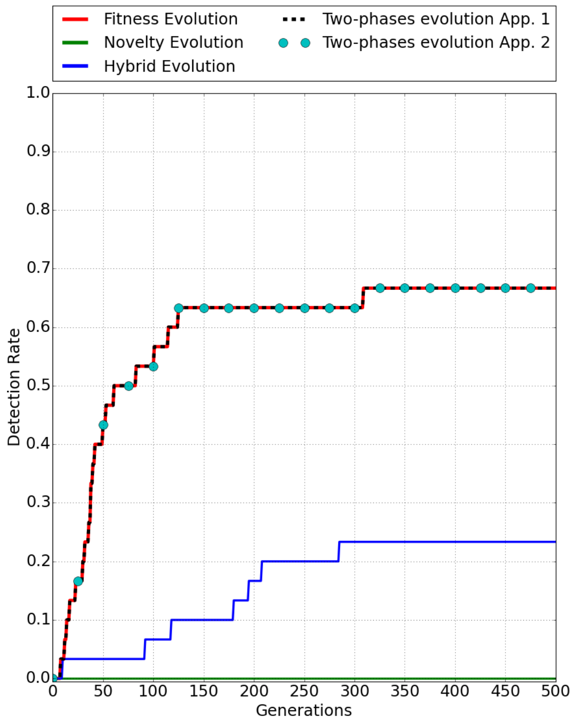


Figure 5.4: Best individual's detection rate along the generations

Observing figure 5.3, the most noticeable fact is that the fitness of the best individual does not increase significantly along the generation, in the novelty-only evolution case. In contrast with this test, the hybrid evolution case achieved higher fitnesses values in later generations by converging towards the fitness value of 0.95 in the 500th generation. The other three cases achieved the same values and their lines are overlapped. Their best fitness achieved a value around 0.97, which mean that the evolution in these three cases resulted in the increase of the fitness towards higher values, when compared to the hybrid case.

Interpreting figure 5.4, the first noticeable result is that the number of objects detected in the novelty case is 0. The hybrid evolution achieves values around 0.25 at the end of the run, which means that in roughly 25% of the runs, the best individual from the last generation was able to contain an object detected by the flower classifier. Concerning the remaining three cases in 5.4 they have overlapped values of detection rate, which are generally higher than the hybrid case. As for figure 5.3, the same three approaches, they have very similar values.

Mainly due to the similar results between the fitness evolution and both two-phases approaches, a new perspective was considered in terms of average detection rate per generation. Thus, figure 5.5 depicts how the average detection rate of each generation varies, instead of considering only the best individual.

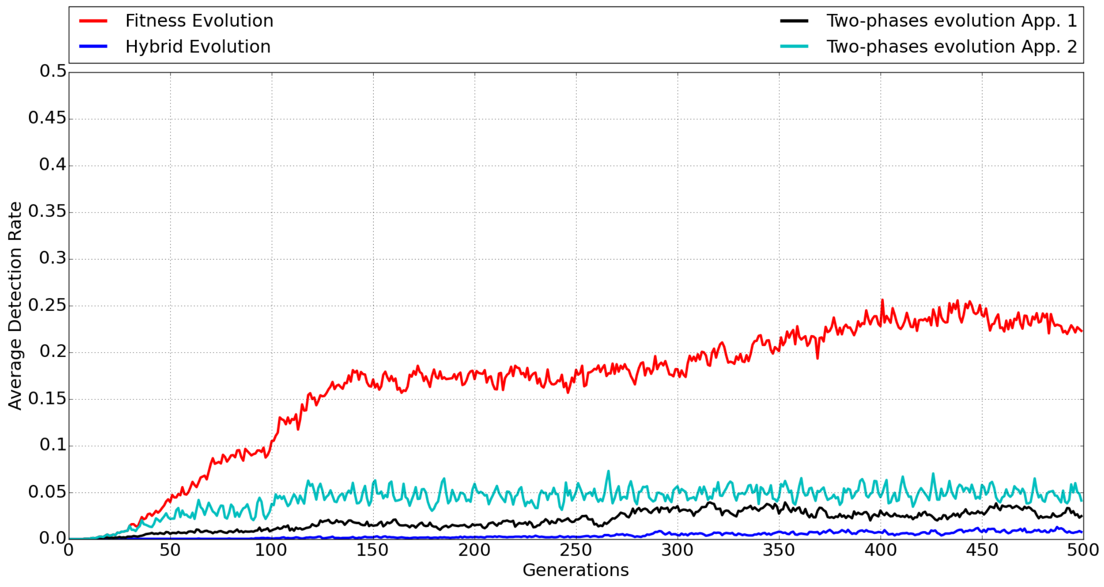


Figure 5.5: Average detection rate along the generations

In this case, it is possible to notice that in all the generations, making a comparative interpretation between the approaches depicted, this fitness-only evolution achieves always the highest values. For the instance, in the last generation, 22% of the population, in average, is detected as containing an object.

The two-phase approach 2 is the approach which achieves the second highest values, with values around 5% during most the execution time of the algorithm. In its turn, the two-phase approach 1 line is always under the last one, with fewer individuals detected as containing an object. Finally, the hybrid approach seems to get values of average detection rate which are a bit higher than 0%.

The second part of these results chooses the two approaches which are able to detect more objects. For the remaining tests one chose the fitness-only evolution and the two-phases evolution approach 2. For the sake of simplicity, in the remainder of this section this last approach will be denominated novelty evolution, because it was the chosen approach, which contains novelty mechanisms, to compare with the fitness evolution.

These two approaches were used to compare the number of images which are able to enter in the archive. Recall that the archive contains images in which the object classifier detected an object and at the same time, achieved an archive novelty value that passed a given threshold. Bearing this idea in mind, figure 5.6 depicts how the archive size varies, in average, along the generations. In this figure, three classifiers were used: a flower classifier, a face classifier and a leaf classifier. Besides, 60 runs were performed for each test.

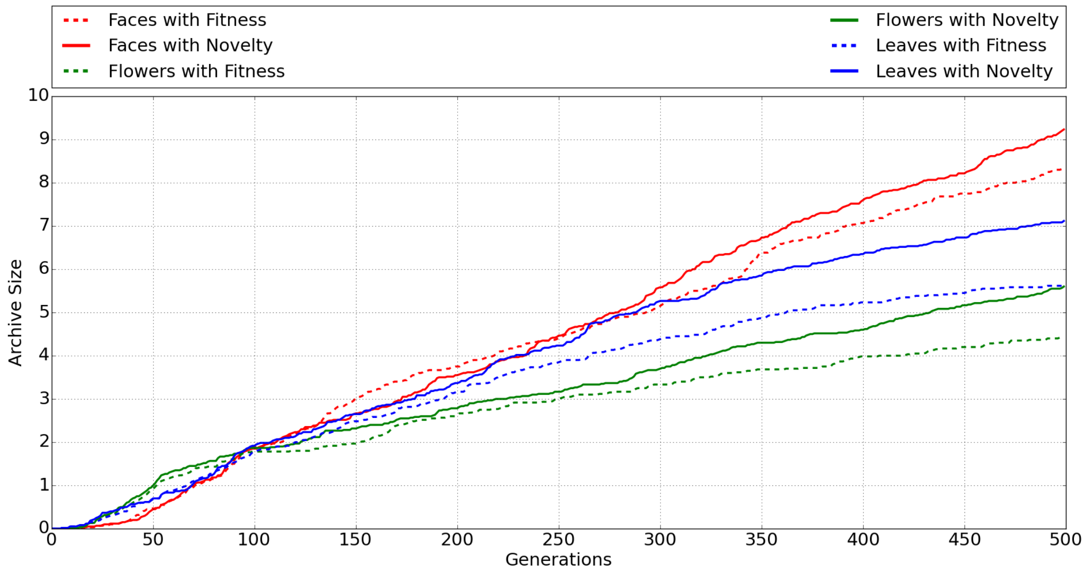


Figure 5.6: Archive size progress throughout the generations evolving figurative images

Two interesting facts can be extracted from figure 5.6. The first one is that the chosen novelty approach is able to insert more images into the archive than the fitness approach, for all the objects tested. Considering the last generation, in faces, novelty achieves in average roughly 9 images against 8 images, regarding fitness. In the leaves case, novelty had 7 images per run against nearly 6 images achieved with fitness evolution. Finally, in the flowers case, novelty had almost 6 images whereas fitness achieved achieved a bit more than 4 images per run.

The second fact extracted from figure 2 is that, comparing for each object fitness and novelty approaches, it is possible to see that the existent gap between both tends to increase as we get newer generations.

In order to assess how many images generated were considered as diverse, the images that entered in the archive, for each run, were submitted to the DBSCAN algorithm [97]. Using RMSE as a distance metric, the more clusters were detected by the algorithm, the more images were considered as diverse. With these results, a pairwise comparison using the same seed for both fitness and the chosen novelty approach was performed. Table 5.2 depicts the results obtained from the pairwise comparison. They present, for each seed, the percentage of runs in which novelty was able to get more diverse images, the same number of images or less diverse images, when compared to fitness. Objects mentioned in bold mean that the results were statistically significant after applying the Wilcoxon statistical test with 95% of confidence. The statistical test was applied to the number of clusters generated by the DBSCAN algorithm.

Table 5.2: Pairwise comparison results between the diverse figurative images generated by fitness and novelty

		Probabilities considering successful runs		
Object	Successful runs	Beneficial	Neutral	Detracted
Faces	73.33%	52.27%	31.82%	15.91%
Flowers	68.33%	43.90%	31.71%	24.39%
Leaves	78.33%	46.81%	38.30%	14.89%

5.4.3 Analysis

From the results presented regarding the first part of the experimentation, namely figures 5.3 and 5.4, it is possible to conclude that the novelty-only approach is in fact a failure because it is not able to promote evolution and it does not detect a single object during the 30 runs. This result is consistent with previous studies about novelty and their problem performing in problems with bigger search spaces.

The hybrid evolution, although it achieves higher detection rates, it evolves with more difficulty and its detection rates are not the highest. The most important factor that contributes to the object detection rate is the elitism. Saving the best individual in terms of fitness assures that the solution which is closer to be considered as containing an object is not lost.

However, a problem persists, because the desired scenario is to obtain diversity considering only the feasible individuals (individuals whose correspondent image contains an object) and hybrid evolution considers novelty taking into account the closest neighbours, whether they contain an object or not. Therefore, hybrid evolution will somehow get lost from the objective of getting images which resemble objects by favouring novelty exclusively.

The other three solutions have the exact same values in figure 5.4 and very similar values in figure 5.3. This fact is again explained with the elitism choice. Maintaining the best individual has the consequence of assuring the value of the best individual.

Therefore, when the best individual contains an object, no matter what type of evolution is used, the use of elitism assures that from that moment, the results will be the same in terms of best individual detection rate. Despite the same fact cannot be generalized to the fitness, it has a big influence and explains the similarity of the three curves.

Given that the results were so similar between those three approaches, the analysis in terms of average detection rate gave a different insight to assess the capability of generating feasible individuals by those three approaches. Therefore, analysing figure 5.5, the main conclusion that can be drawn is that the hybrid mechanism, the more time it is applied, the more probable it is to get lost in zones of the search space which do not contain feasible individuals.

This thought led to the idea that what matter may not be the number of individuals *per se*, but instead assessing the capability of generating different feasible individuals when they appear. Bearing this idea in mind, 60 runs were executed using the most successful novelty mechanism and the fitness evolution.

Analysing figure 5.6, the results are clear, because no matter the desired object chosen, the novelty mechanism will generate more feasible ans distinct individuals than the fitness evolution. The interpretation of these results improved results was the following. Hybrid evolution favours the appearance of new individuals but the probability of moving towards infeasible regions of the search space is high. On the other hand, fitness evolution can converge to feasible zones of the search space. However, its probability of generating diverse content is lower (one used the word converge to characterize how fitness evolves). Therefore, one uses fitness evolution to get towards feasible zones as quick as possible. When a certain number of individuals is classified as containing an object (in this case, 15), the evolution type switches to hybrid, in order to work using novelty as one of the criteria. Using this method increases the probability of getting more images into the archive than fitness, but at the same time, increases the probability of moving towards infeasible search spaces. For the instance, when less than 5 individuals are detected as containing an object, it a sign that fitness needs to be adopted as the evolution method, in order to evolve again towards feasible zones of the search space, whether they are the same or not.

In fact, analysing table 5.2, the pairwise comparison showed that, with this novelty mechanism, it is more likely to improve in terms of the number of diverse images than degrading. The probability of degrading is considerably low, and the chances of getting at least the same number of images is very high ($> 70\%$).

5.5 Experimentation - Evolving ambiguous images

This experimentation covers the use of the same novelty mechanism, discovered in section 5.4, but applied to the evolution of ambiguous images. Recall that the novelty mechanism adopted consists in switching between a fitness evolution and an hybrid evolution, according to the number of individuals which resemble a given object. To assess novelty capabilities of promoting diversity in ambiguous images, one used the same set of classifiers utilized in the experimentation done in chapter 4 (namely in section 4.3). Therefore, this study involves the use of two pairs of classifiers: a robust pair of faces and leaves classifiers and a permissive pair of faces and leaves classifiers.

The relevant parameters used in this experimentation are presented in section 5.5.1. Then, the results of a direct comparison between fitness and novelty mechanisms using the two pairs of classifiers are presented in section 5.5.2. These results are then analysed in section 5.5.3.

5.5.1 Setup

This parameter setup shares similarities with the experimentation performed in chapter 4 and the experimentation performed in section 5.4. In the one hand, as this study addresses the evolution of ambiguous images, one used the sets of classifiers when compared to chapter 4: a pair of robust classifiers (faces and leaves) and a pair of permissive classifiers (faces and leaves). On the other hand, this study addresses the introduction of novelty. For this reason, this experimentation focuses on the comparison between fitness and novelty mechanisms in terms of quantity and diversity of the images.

Once again, the relevant parameters are explained by breaking them down in different components.

Classifiers

As it was referred in section 5.5.1, two sets of classifiers are used in this study. As the classifiers influence the fitness function and this study follows the steps of the experimentation done in chapter 4, the same fitness functions are used.

Object Detection Algorithm

The parameters concerning the object detection algorithm are same as the ones used in chapter 4. For further details, consult section 4.3.1.

Evolutionary Algorithm

Concerning the evolutionary algorithm, the parameters chosen are a crossover between previous studies. They are described in table 5.3.

Table 5.3: GP engine parameters used

Parameter	Setting
Population Size	100
Generations	1000
Crossover Probability	0.8
Mutation probability	0.05
Initialization Method	Ramped Half-and-Half
Initial Maximum Depth	5
Mutation max tree depth	3
Elite size	1
Tournament Size	5
max parameter	5
Dissimilarity Metric	Root Mean Squared Error
Archive Threshold	60

Briefly explaining, all the parameters adopted from chapter 4 are the same, for the sake of consistency. Regarding the parameters added due to the novelty mechanisms, the majority of the parameters were also maintained. However, as it is predicted that the number of distinct images produced would be lower, the archive threshold was reduced, in order to accept the introduction of more images.

5.5.2 Results

The results presented in this section cover the comparison between the two approaches chosen in the previous section: the fitness one and the novelty one. Instead of using a single classifier, two pairs of classifiers are used, and for each pair, 60 runs are performed.

In similarity with the previous section, the results are presented in terms of variation of the archive size along the generations and they can be observed in figure 5.7.

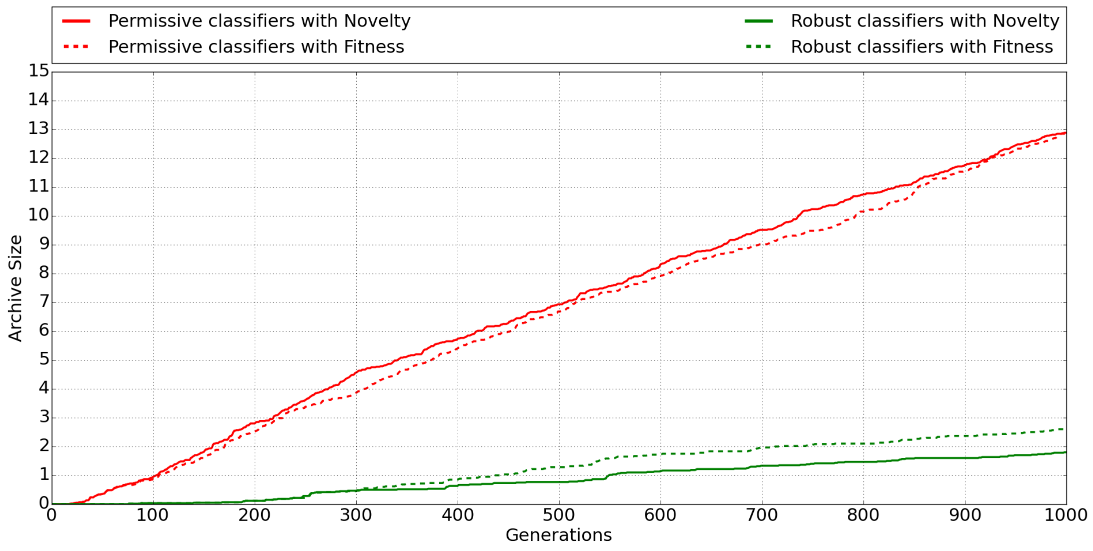


Figure 5.7: Archive size progress throughout the generations evolving ambiguous images

Looking to that chart, and observing only the green lines, which refer to the robust set of classifiers, it is possible to notice that the fitness approach is able to insert more images to the archive, in average, than the novelty approach, throughout the generations. This phenomenon contrasts with the facts detected in the previous section, where novelty achieved more images in the archive than fitness.

As for the permissive set, the number of images in the archive seems to be higher in the novelty case. However, the gap between novelty and fitness does not seem to increase in newer generations. Moreover, the number of images generated using permissive classifiers are considerably higher than the case where robust classifiers are used.

Similarly to previous section, the final archive contents were submitted to the DBSCAN algorithm clustering algorithm, seed by seed. Then, the percentages of occurrences in which novelty improves, degrades, or runs with the same performance as the fitness approach is traced, and their values are presented in table 5.4. The Wilcoxon test was applied again using the number of images in the archive, but this time, the results were not statistically significant.

5.5.3 Analysis

The results obtained in the previous section can be analysed from two different perspectives. Comparing the sets of classifiers used, and using figure 5.7 to analyse, it is clear that the types of classifiers used influence the number of diverse images

Table 5.4: Pairwise comparison results between the diverse ambiguous images generated by fitness and novelty

		Probabilities considering successful runs		
Classifiers set	Successful runs	Beneficial	Neutral	Detracted
Robust Set	41.67%	0.00%	88.00%	12.00%
Permissive Set	80.00%	41.67%	25.00%	33.33%

generated. The higher number of images in the permissive set can be interpreted in the following way. The classifiers do not influence the search space size, but they influence in the choice and distinction between feasible and infeasible zones. Thus, robust classifiers tend to restrict the search space in the sense that finding feasible individuals in a harder task. With restricted search spaces, it is harder to find images and mainly, to find images which can be considered as diverse. This analysis can also be made in terms of comparison fitness vs. novelty.

Observing figure 5.7 and table 5.4, it seems that the classifiers' robustness influence the effectiveness of novelty mechanism. With robust classifiers the search space gets restricted, and these constraints are consequently incompatible with the employment of novelty techniques. On the other hand, permissive classifiers tend to have larger feasible zones of the search space, where novelty can in fact act, and increase the probability of getting better results.

5.6 Synthesis

In this chapter one extended previous norBErT approaches, in order to promote a different type of evolution where the uniqueness of each solution is valued instead of considering (only) the fitness function. This study was performed, in order to pursue the goal of getting a higher range of distinct images.

The novelty mechanism was employed in norBErT by adding a component and modifying an existent one. In order to filter which images could be considered as diverse, it was added an novelty archive component which was responsible to hold the images which was considered as containing an object by the object classifier(s) and at the same time, achieved an archive novelty value higher than a given threshold. This value was computed using a dissimilarity metric between the candidate image to enter in the archive and the ones that were already there. The modified component was the tournament selection. It was adapted to select the best candidates based on fitness, evolving novelty or both. Evolving novelty is calculated using a dissimilarity between each image and the nearest neighbours selected from the tournament and the archive.

The results showed that, in the case of evolving figurative images, novelty mechanisms outperforms mechanisms based on fitness only, in terms of number of images considered as distinct from the others. Nevertheless, when making a pairwise comparison using running the algorithm with the same seed number, there is a small inherent risk of degrading the results instead of improving them. This risk is higher when one tries to use novelty mechanisms in the evolution of ambiguous images, and it depends on the characteristics of the classifier: robust classifiers shape the search space to be more constrained, which in its turn, complicates the task of finding a large number of diverse images when the search space size is large. On the other hand, permissive classifiers allow the appearance of more diverse images and is are more compatible with the employment of novelty mechanisms.

Chapter 6

Conclusions and Future Work

The field of evolutionary art is rooted in the seminal works of Dawkins [69] and Sims [2], who established the foundations for subsequent approaches.

Correia and Machado [5] used a general purpose approach which they proven to be able to evolve figurative images of any object desired. Their approach (or more exactly, the object desired to evolve) depended solely on an object classifier, which was used to help assigning fitness to the images.

In the current work, three main objectives were achieved. First, based on the work of Correia and Machado, their method was reimplemented in order to allow the evolution of figurative images with a set of classifiers created using datasets built manually. Then, this work was adapted to perform evolution taking into account multiple classifiers, in order to enable the evolution of ambiguous images. Finally, the method was extended in order to allow the generation of more images containing the desired object(s) and at the same time distinct, by employing novelty search mechanisms.

In this work, a tool called as norBErT was created, in order to attain the objectives defined. This tool generates images using an evolutionary algorithm, using single or multiple object classifiers to help in the fitness assignment task, and dissimilarity metrics to calculate uniqueness values.

The development of such a tool involved the study of the state of the art in Evolutionary Computation (namely genetic programming and multiobjective evolutionary algorithms), Evolutionary Art and mechanisms which are able to improve diversity in Evolutionary Computation.

From the experiments performed, the approach adopted was able to evolve figurative images that resemble faces, face silhouettes, flowers and leaves. However, it should be noted that what is computationally or humanly considered as an object is different sometimes. This fact is even more visible when one tries to evolve ambiguous images, as the evolution of these type of images requires the same image to be evaluated by several classifiers (in this case leaves and faces) instead of just one. The approach is able to evolve ambiguous images of leaves and faces, and these ambiguous images can be recognized by humans, identifying both objects at the same time.

Regarding the employment of novelty techniques, the novelty mechanism developed consisted in changing between a fitness-only evolution and an hybrid evolution, depending on the percentage of feasible individuals.

It was concluded that the constraints in the search space play an important role on the effectiveness of applying novelty search. Search spaces which are very constrained are not favourable to the employment of novelty search techniques. On the other hand, permissive classifiers (or at least using less classifiers to guide evolution) are compatible with the use of novelty search. In these cases, the novelty mechanism adopted outperformed the fitness based evolution in terms of number of diverse images generated, although they have an associated risk of, sometimes, degrading the results instead of improving them.

Given this drawback, future work can build on this study and attempt to lower the risk of degrading the results using novelty mechanisms. To study this subject, it would be important to assess in what conditions this deterioration happens, and use the relevant data to find a different criterion to switch between novelty and fitness evolutions, or develop an adaptive scheme of switching evolution types. Another interesting topic to tackle is the exploration of the classifier component. Training these classifiers involves choosing a good dataset and a lot of parameters. Therefore, it would be interesting to explore the tuning of these parameters through self evolutionary means. Another interesting topic to explore is the way fitness functions from different classifiers are aggregated. Only one was chosen for this work, but it would be interesting to assess whether the final ambiguous images generated would benefit using a paretto-based approach.

Bibliography

[1] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, USA, 1992.

[2] Karl Sims. Artificial evolution for computer graphics. *SIGGRAPH Comput. Graph.*, 25(4):319–328, July 1991.

[3] Jon McCormack. Facing the future: Evolutionary possibilities for human-machine creativity. In Juan Romero and Penousal Machado, editors, *The Art of Artificial Evolution*, Natural Computing Series, pages 417–451. Springer Berlin Heidelberg, 2008.

[4] P. Machado and A. Cardoso. All the truth about NEvAr. *Applied Intelligence, Special Issue on Creative Systems*, 16(2):101–119, 2002.

[5] Penousal Machado, João Correia, and Juan Romero. Improving face detection. In Alberto Moraglio, Sara Silva, Krzysztof Krawiec, Penousal Machado, and Carlos Cotta, editors, *Genetic Programming – 15th European Conference, EuroGP 2012, Málaga, Spain, April 11-13, 2012. Proceedings*, volume 7244 of *Lecture Notes in Computer Science*, pages 73–84. Springer, 2012.

[6] Paul A. Viola and Michael J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR (1)'01*, pages 511–518, 2001.

[7] Charles Darwin. *On the Origin of Species by Means of Natural Selection.* Murray, London, 1859. or the Preservation of Favored Races in the Struggle for Life.

[8] Agoston E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing.* SpringerVerlag, 2003.

[9] Ernesto Costa and Anabela Simões. *Artificial Intelligence: foundatons and applications (in portuguese).* FCA, 3rd edition, 2008.

[10] Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors. *Evolutionary Computation 1: Basic Algorithms and Operators.* Dirac House, Temple Back, Bristol, UK: Institute of Physics Publishing Ltd. (IOP), January 2000.

[11] Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors. *Evolutionary Computation 2: Advanced Algorithms and Operators.* Dirac House, Temple Back, Bristol, UK: Institute of Physics Publishing Ltd. (IOP), November 2000.

[12] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* MIT Press, Cambridge, MA, USA, 1992.

[13] J. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM Journal on Computing*, 2(2):88–105, 1973.

[14] Kenneth Alan De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, Ann Arbor, MI, USA, 1975. AAI7609381.

[15] Wolfgang Banzhaf, Frank D. Francone, Robert E. Keller, and Peter Nordin. *Genetic Programming: An Introduction: on the Automatic Evolution of Computer Programs and Its Applications.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[16] John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs.* MIT Press, Cambridge, MA, USA, 1994.

[17] Riccardo Poli, William B. Langdon, and Nicholas Freitag McPhee. *A Field Guide to Genetic Programming.* Lulu Enterprises, UK Ltd, 2008.

[18] Heikki Maaranen, Kaisa Miettinen, and Antti Penttinen. On initial populations of a genetic algorithm for continuous optimization problems. *Journal of Global Optimization*, 37(3):405–436, 2007.

[19] T. Furuhashi, Y. Miyata, K. Nakaoka, and Y. Uchikawa. A new approach to genetic based machine learning and an efficient finding of fuzzy rules. In Takeshi Furuhashi, editor, *Advances in Fuzzy Logic, Neural Networks and Genetic Algorithms*, volume 1011 of *Lecture Notes in Computer Science*, pages 173–189. Springer Berlin Heidelberg, 1995.

[20] Inman Harvey. The microbial genetic algorithm, 1996.

[21] Peter Smith and Dr. Peter Smith. Finding hard satisfiability problems using bacterial conjugation, 1996.

[22] Peter Smith and Dr. Peter Smith. Conjugation - a bacterially inspired form of genetic recombination. In *Stanford University*.

[23] A. Simões and E. Costa. Transposition: A biological-inspired mechanism to use with genetic algorithms. In *Artificial Neural Nets and Genetic Algorithms*, pages 178–186. Springer Vienna, 1999.

[24] Nichael Lynn Cramer. A representation for the adaptive generation of simple sequential programs. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 183–187, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc.

[25] Matthew J Streeter. The root causes of code growth in genetic programming. In *Genetic programming*, pages 443–454. Springer, 2003.

[26] Mihai Oltean, Crina Grosan, Laura Diosan, and Cristina Mihaila. Genetic programming with linear representation: a survey. *International Journal on Artificial Intelligence Tools*, 18(2):197–238, 2009.

[27] Riccardo Poli. Parallel distributed genetic programming. Technical report, SCHOOL OF COMPUTER SCIENCE, UNIVERSITY OF BIRMINGHAM, 1999.

[28] Astro Teller. Advances in genetic programming. chapter Evolving Programmers: The Co-evolution of Intelligent Recombination Operators, pages 45–68. MIT Press, Cambridge, MA, USA, 1996.

[29] R Sivaraj and T Ravichandran. A review of selection methods in genetic algorithm. *International journal of engineering science and technology*, 3(5):3792–3797, 2011.

[30] John R Koza and Riccardo Poli. Genetic programming. In *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pages 127–164. 2005.

[31] Peter J Angeline. Subtree crossover: Building block engine or macromutation. *Genetic Programming*, 97:9–17, 1997.

[32] Carlos M Fonseca and Peter J Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary computation*, 3(1):1–16, 1995.

[33] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *evolutionary computation, IEEE transactions on*, 3(4):257–271, 1999.

[34] Wilfried Jakob, Martina Gorges-Schleuter, and Christian Blume. Application of genetic algorithms to task planning and learning. In *Parallel Problem Solving from Nature 2, PPSN-II, Brussels, Belgium, September 28-30, 1992*, pages 293–302, 1992.

[35] F. Gembicki and Y.Y. Haimes. Approach to performance and sensitivity multiobjective optimization: The goal attainment method. *Automatic Control, IEEE Transactions on*, 20(6):769–771, Dec 1975.

[36] Michael P Fourman. Compaction of symbolic layout using genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 141–153. L. Erlbaum Associates Inc., 1985.

[37] J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh, PA, USA, July 1985*, pages 93–100, 1985.

[38] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[39] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast elitist multi-objective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2000.

[40] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. Multiobjective optimization using the niched pareto genetic algorithm. Technical report, 1994.

[41] Deepti Gupta and Shabina Ghafir. An overview of methods maintaining diversity in genetic algorithms.

[42] Samir W. Mahfoud. Niching methods for genetic algorithms. Technical report, 1995.

[43] Anikó Ekárt and SandorZ. Németh. Maintaining the diversity of genetic programs. In JamesA. Foster, Evelyne Lutton, Julian Miller, Conor Ryan, and Andrea Tettamanzi, editors, *Genetic Programming*, volume 2278 of *Lecture Notes in Computer Science*, pages 162–171. Springer Berlin Heidelberg, 2002.

[44] Eelco den Heijer and AE Eiben. Maintaining population diversity in evolutionary art. In *Evolutionary and Biologically Inspired Music, Sound, Art and Design*, pages 60–71. Springer, 2012.

[45] Joel Lehman and Kenneth O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI*. MIT Press, 2008.

[46] David E. Goldberg. Simple genetic algorithms and the minimal, deceptive problem. In Lawrence Davis, editor, *Genetic algorithms and simulated annealing*, Research Notes in Artificial Intelligence, pages 74–88. Pitman, London, 1987.

[47] Reiko Tanese. Distributed genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 434–439, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[48] David Goldberg, K. Deb, and B. Korb. Messy genetic algorithms: motivation, analysis, and first results. *Complex Systems*, (3):493–530, 1989.

[49] L. Darrell Whitley. Fundamental principles of deception in genetic search. In *Foundations of Genetic Algorithms*, pages 221–241. Morgan Kaufmann, 1991.

[50] David E Goldberg, Kalyanmoy Deb, and Bradley Korb. Messy genetic algorithms revisited: Studies in mixed size and scale. *Complex Systems*, 4:415–444, 1990.

[51] Rob Saunders and John S. Gero. The digital clockwork muse: A computational model of aesthetic evolution. In *The AISB'01 Symposium on AI and Creativity in Arts and Science, SSAISB*, pages 12–21, 2001.

[52] C. Martindale. *The Clockwork Muse: The Predictability of Artistic Change.* BasicBooks, 1990.

[53] Avi Silberschatz and Alexander Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Trans. on Knowl. and Data Eng.*, 8(6):970–974, December 1996.

[54] Joel Lehman and Kenneth O. Stanley. Efficiently evolving programs through the search for novelty. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pages 837–844, New York, NY, USA, 2010. ACM.

[55] Antonios Liapis, Héctor P. Martínez, Julian Togelius, and Georgios N. Yannakakis. Transforming exploratory creativity with delenox.

[56] Antonios Liapis, Georgios N Yannakakis, and Julian Togelius. Constrained novelty search: A study on game content generation. *Evolutionary Computation*, 2014.

[57] Joel Lehman, Sebastian Risi, and Kenneth O. Stanley. On the benefits of divergent search for evolved representations. In *Proceedings of the EvoNet 2012 Workshop at ALIFE XIII*, 2012.

[58] Joel Lehman and Kenneth O. Stanley. Revising the evolutionary computation abstraction: Minimal criteria novelty search. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pages 103–110, New York, NY, USA, 2010. ACM.

[59] Antonios Liapis, Georgios Yannakakis, and Julian Togelius. Enhancements to constrained novelty search: Two-population novelty search for generating game content. In *Proceedings of Genetic and Evolutionary Computation Conference*, 2013.

[60] Steven Orla Kimbrough, Gary J Koehler, Ming Lu, and David Harlan Wood. Introducing a feasible-infeasible two-population (fi-2pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch. Technical report, March 14, 2005.(Working Paper), 2005.

[61] Giuseppe Cuccu and Faustino Gomez. When novelty is not enough. In Cecilia Di Chio, Stefano Cagnoni, Carlos Cotta, Marc Ebner, Anikó Ekárt, AnnaI. Esparcia-Alcázar, JuanJ. Merelo, Ferrante Neri, Mike Preuss, Hendrik Richter, Julian Togelius, and GeorgiosN. Yannakakis, editors, *Applications of Evolutionary Computation*, volume 6624 of *Lecture Notes in Computer Science*, pages 234–243. Springer Berlin Heidelberg, 2011.

[62] David E. Goldberg and Jon Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, pages 41–49, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc.

[63] Geoffrey E Hinton and Richard S Zemel. Autoencoders, minimum description length, and helmholtz free energy. *Advances in neural information processing systems*, pages 3–3, 1994.

[64] Yves Chauvin and David E. Rumelhart, editors. *Backpropagation: Theory, Architectures, and Applications.* L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1995.

[65] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[66] Casey Chesnut. Self organizing map ai for pictures. `http://www.generation5.org/content/2004/aiSomPic.asp`, 11 2004.

[67] Andrew Rowbottom. What is evolutionary art? `http://www.netlink.co.uk/~snaffle/form/evolutio.html`, August 1998.

[68] Matthew Lewis. Evolutionary visual art and design. In Juan Romero and Penousal Machado, editors, *The Art of Artificial Evolution*, Natural Computing Series, pages 3–37. Springer Berlin Heidelberg, 2008.

[69] R Dawkins. *The Blind Watchmaker.* Longman Scientific and Technical, 1986.

[70] L World. Aesthetic selection: The evolutionary art of steven rooke [about the cover]. *Computer Graphics and Applications, IEEE*, 16(1):4, 1996.

[71] Tatsuo Unemi. SBART 2.4: breeding 2D CG images and movies and creating a type of collage. In *Third International Conference on Knowledge-Based Intelligent Information Engineering Systems, KES 1999*, pages 288–291, Adelaide, Australia, 31 August-1 September 1999. IEEE.

[72] Penousal Machado and Amílcar Cardoso. Nevar–the assessment of an evolutionary art tool. *Proceedings of the AISB00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science, Birmingham, UK*, 456, 2000.

[73] David Hart. D. a. hart abstract art. `http://www.dahart.com/`, 2006.

[74] Ellie Baker and Margo Seltzer. Evolving line drawings. In *PROCEEDINGS OF THE FIFTH INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS*, pages 91–100. Morgan Kaufmann Publishers, 1994.

[75] Hideyuki Takagi. Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, 2001.

[76] Scott Draves. The electric sheep screen-saver: A case study in aesthetic evolution. In *Applications of Evolutionary Computing*, pages 458–467. Springer, 2005.

[77] Rikard Gatarski. Evolutionary banners: An experiment with automated advertising design. In *Conference on Telecommunications and Information Markets (COTIM'99)*, 1999.

[78] JI Van Hemert and MLM Jansen. An engineering approach to evolutionary art. In *Gecco-2001: Proceedings of the Genetic and Evolutionary Computation Conference: a Joint Meeting of the Sixth Annual Genetic Programming Conference (GP-2001) and the Tenth International Conference on Genetic Algorithms (ICGA-2001): July 7-11, 2001, San Francisco, California*, volume 17, page 177. Citeseer, 2001.

[79] Penousal Machado, Juan Romero, Amílcar Cardoso, and Antonino Santos. Partially interactive evolutionary artists. *New Generation Computing*, 23(2):143–155, 2005.

[80] Gary R Greenfield et al. Color dependent computational aesthetics for evolving expressions. In *Bridges: Mathematical Connections in Art, Music, and Science*, pages 9–16. Bridges Conference, 2002.

[81] B.J. Ross, W. Ralph, and Hai Zong. Evolutionary image synthesis using a model of aesthetics. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 1087–1094, 2006.

[82] GaryR. Greenfield. Co-evolutionary methods in evolutionary art. In Juan Romero and Penousal Machado, editors, *The Art of Artificial Evolution*, Natural Computing Series, pages 357–380. Springer Berlin Heidelberg, 2008.

[83] Jon McCormack. Open problems in evolutionary music and art. In *Applications of Evolutionary Computing*, pages 428–436. Springer, 2005.

[84] Steve DiPaola and Liane Gabora. Incorporating characteristics of human creativity into an evolutionary art algorithm. *Genetic Programming and Evolvable Machines*, 10(2):97–110, 2009.

[85] Penousal Machado, Juan Romero, Bill Manaris, Antonino Santos, and Amílcar Cardoso. Power to the critics — A framework for the development of artificial art critics. In *IJCAI 2003 Workshop on Creative Systems*, Acapulco, Mexico, 2003.

[86] Penousal Machado, João Correia, and Juan Romero. Expression-based evolution of faces. In *Evolutionary and Biologically Inspired Music, Sound, Art and Design – First International Conference, EvoMUSART 2012, Málaga, Spain, April 11-13, 2012. Proceedings*, volume 7247 of *Lecture Notes in Computer Science*, pages 187–198. Springer, 2012.

[87] João Correia, Penousal Machado, Juan Romero, and Adrián Carballal. Evolving figurative images using expression-based evolutionary art. In *Proceedings of the fourth International Conference on Computational Creativity (ICCC)*, pages 24–31, 2013.

[88] Sean Luke, Liviu Panait, Gabriel Balan, Sean Paus, Zbigniew Skolicki, Jeff Bassett, Robert Hubley, and A Chircop. Ecj: A java-based evolutionary computation research system. *Downloadable versions and documentation can be found at the following url: http://cs. gmu. edu/eclab/projects/ecj*, 2006.

[89] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[90] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *IEEE ICIP 2002*, pages 900–903, 2002.

[91] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29(1):51–59, 1996.

[92] Yucui JU, Hua ZHANG, and Yanbing XUE. Research of feature selection and comparison in adaboost based object detection system. *Journal of Computational Information Systems*, 9(22):8947–8954, 2013.

[93] Hannes Caspar, Kerem Ergun, and Martin Wunderwald. Facity project website. `http://www.facity.com/`, 2010.

[94] João Correia, Penousal Machado, and Juan Romero. Improving haar cascade classifiers through the synthesis of new training examples. In Terence Soule and Jason H. Moore, editors, *Genetic and Evolutionary Computation Conference, GECCO '12, Philadelphia, PA, USA, July 7-11, 2012, Companion Material Proceedings*, pages 1479–1480. ACM, 2012.

[95] Martina Jakesch, Helmut Leder, and Michael Forster. Image ambiguity and fluency. *PLoS ONE*, 8(9):e74084, 09 2013.

[96] Jean-Baptiste Mouret. Novelty-based multiobjectivization. In *New Horizons in Evolutionary Robotics*, pages 139–154. Springer Berlin/Heidelberg, 2011.

[97] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.