Master in Informatics Engineering
Dissertation
Final Report

Evolutionary Computation for Assessing and Improving Classifier Performance

João Nuno Gonçalves Costa Cavaleiro Correia
jncor@student.dei.uc.pt

Advisor:
Penousal Machado
Date: September 1, 2011

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Abstract

In this dissertation we explore the use of Evolutionary Computation for assessing and improving the performance of classifier systems, focusing on image classification tasks. A boosting framework for classifier improvement is proposed. The approach relies on the use of an evolutionary computation engine to exploit the potential weaknesses of the classifiers, evolving instances that produce classification errors. Subsequently these instances are become part of the training sets in order to circumvent the identified weaknesses. The framework was instantiated and tested in two image classification scenarios and several validation experiments were conducted. The experimental results attained are described and analyzed. Overall the results show the viability of the proposed approach and indicate future research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Computer Vision is the science that gives the machines the capability to *see*, where *see* is the ability to extract information from an image, process and analyze it in order to solve vision related tasks. From a technological point of view, Computer Vision seeks to apply its theories and models to the construction of bio-inspired computer vision systems [4].

Computer vision systems have evolved from the typical pattern recognition applications and image processing techniques to advanced applications of image understanding, model-based vision, knowledge based vision, and systems that exhibit learning capability. The ability to reason and to learn are the two major capabilities associated with these systems. Through all these years, hypothesis, theories, algorithms and practical applications have been created in the field of Computer Vision and Pattern Recognition, resulting in new techniques of representation, adaptation and learning. However, these processes of generalization, abstraction and learning, represent a challenging frontier for Computer Vision. In order to attend for this fact, a synergy between Computer Vision (CV) and Machine Learning (ML) emerged. The majority of Computer Vision systems include (in some part or in some way) a process of understanding and learning from information, using it to complete the objective of the system [72, 76, 86, 69, 11].

As the research on these areas advances the problems tackled become more complex and so do state of the art approaches, which currently often involve several layers of decision and control parameters. The fine-tuning of the approaches becomes crucial for attaining competitive results. Additionally, in example-based learning approaches, the quality (e.g. completeness, representativeness) of the employed datasets is crucial not only for attaining competitive performances but also for correctly assessing the strengths and shortcomings of the approaches. As Such, developing adequate datasets for training, testing and validation is a complex and time-consuming process, but also a crucial one.

Evolutionary Computation (EC) has been introduced in recent works, imbuing optimization and learning power to the methods utilized, through its main characteristic, evolution. EC is a branch of Artificial Intelligence inspired by the natural process of evolution described by Darwin [10], which attempts to

solve problems by evolving solutions to them. The key concept of the Evolutionary Algorithms (EAs) are: (i) Selection – individuals that are better suited to their environment have higher probabilities of surviving and reproducing; (ii) Inheritance – the descendants inherit characteristics from the progenitors; (iii) Variation – The descendants are not exact copies of the parents. Inheritance and variation occur at the genotype level. By inheriting parts of the genetic code of the progenitors the descendants also tend to inherit some of their characteristics. Variation is attained by the recombination of the genetic code of the progenitors, which is thought to promote the exploitation of characteristics that are already present in the population, and by the introduction of random changes in the genetic code (i.e. mutations) which have the potential to introduce new genetic code, and hence are thought to promote exploration.

EAs have been used in various contexts in CV and ML. In the CV context some examples of its usage are digital filters tuning, parameter optimization and image generation. In ML, EAs have been used to evolve solutions for classifiers parameters, thresholds, feature selection for classification, the classifier itself, among others. Works such as [84, 42, 7, 73] combine EC, CV and ML aspects.

In preliminary experiments, conducted in the course of this dissertation, we combined an evolutionary expression-based image generation system [54] and a state of the art face detection system [86], with the goal of evolving images that resembled human faces. The experiment was a success, in the sense that the evolutionary engine easily evolved hundreds of images that were identified as faces by the classifier. It was a failure, in the sense that most of these images, albeit classified as faces, did not resemble faces to the human eye. Although the results were both unexpected and disappointing they gave us interesting insights.

The EA was able to find and exploit shortcomings of the classifier to "artificially" increase fitness. The propensity of EAs to find "shortcuts" that exploit weaknesses of the fitness assignment scheme is well-known (see, e.g., [81, 84, 59]). This open two possibilities: (i) Using EA approaches to test the robustness of face detector systems; (ii) use the misclassified instances to improve the datasets used in their training, hence improving the face detection system. These ideas can, potentially, be generalized to other classification and identification tasks.

Following these ideas, the present dissertation concerns the use of EC to assess and improve classifier performance through the synthesis of new training, testing and validation examples.

In order to fulfil these goals, we will attend to several objectives:

- Create a framework for classifier assessment and improvement through evolutionary techniques;
- Perform tests with the framework in different classification scenarios and classifiers;
- Study and develop appropriate fitness assignment schemes for each scenario;
- Retrieve and analyze the individuals resulting from the evolutionary runs for the purpose of identifying the shortcomings of the classifiers and improve their performance.
- Assess the viability and utility of the proposed framework;

The creation of novel CV algorithms, ML approaches and evolutionary paradigms is outside the scope of this thesis. Additionally, we will focus exclusively on image classification tasks. We will resort, to a well established evolutionary expression-based image generation engine [54], which will be integrated with image classifiers and adapted, namely in terms of fitness assignment, in order to to suit our goals. To promote the generalization of the attained experimental results we will resort to state of the art classification techniques.

The reminder of the document is divided in seven chapters.

Chapter 2 gives a overview of the research that as been done so far in terms of evolutionary algorithms, classification tasks and face detection.

In Chapter 3 the explanation of the preliminary work is performed. This chapter Covers an evolutionary approach to evolve faces (3.1) and its contribution as the first step for the proposed framework.

A description of the framework is given (4).

The first test to the framework is described in Chapter 5 and refers to a work inspired by Machado et. al.[59] to improve a Style-Based Image classifier. Chapter 6 describes the second test to the framework by exploring an experiment with a Face detection classifier.

Chapter 7 describes the conclusions drawn from the research done for this dissertation and indicates future research.

Chapter 2

State of the Art

This dissertation is an interdisciplinary work that combines areas such as CV, ML and EC. The current chapter covers theoretical and practical aspects of some disciplines within the mentioned areas, which contribute for the objectives of the dissertation. First, in section 2.1, an introduction to Evolutionary Computation and its sub-areas is presented. The application of EC to the field of image generation is discussed in section 2.1.2. Section 2.2 makes an overview the classifier system employed in image classification tasks. Finally, in section 2.3, a survey of face detection algorithms is presented.

2.1 Evolutionary Computation

EC is a field of Artificial Intelligence that makes use of a variety of evolutionary computational models inspired by the Darwinian [10] principles of the natural process of evolution for problem solving. These computational models are usually referred to as EAs.

Over the decades, and with the advances in computer technology, a growing demand for problem-solving automation has been created. In order to attend to that matter, EAs have been frequently used as problem solving tools. The extension of EAs application reach areas outside the traditional Computer Science scope. Examples of their application include: data-mining [3], communications [39], robotics [93], games [65], medicine [91] and finance [82].

Despite the differences between EA approaches they share common concepts and methods. Their basis for problem solving is a stochastic population-based optimization algorithm.

Figure 2.1 presents a generic EA. The population is usually initialized by a stochastic method. Evaluation is the process of fitness assignment of the individuals, that is context and problem dependent. Selection picks the parents for the next population. By exchanging information between the selected parents (recombination) new children are created and then can be further perturbed by mutation. Finally the survive step determines who persists for the next population.

```
procedure EA; {
        t = 0;
        initialize population P(t);
        evaluate P(t);
        until (done) {
                t = t + 1;
                parent_selection P(t);
                recombine P(t);
                mutate P(t);
                evaluate P(t);
                survive P(t);
        }
}
```

Figure 2.1: Generic Evolutionary Algorithm [80]

2.1.1 Sub-areas of Evolutionary Algorithms

Four different EA sub-areas are acknowledged: Genetic Algorithms (GA), Evolution Strategies (ES), Genetic Programming (GP) and Evolutionary Programming (EP) [88]. Although similar at the highest level, there are differences in aspects such as: representation of individuals, selection mechanisms, genetic operators and performance measure.

Although the focus of the current dissertation proposal is GP (section 2.1.1), the different sub-areas of Evolutionary Algorithms, GA (section 2.1.1), ES (section 2.1.1) and EP (section 2.1.1), are briefly described in the reminder of this subsection.

Evolutionary Strategies

ESs were developed in Germany, in the late 1960's and 1970's by Ingo Rechenberg and Hans-Paul Schwefel [88]. ESs have historically been used in real-valued representations of optimization problems. This approach emphasizes mutation over recombination. Another particular aspect of this approach is the different ways of manipulating parents and offsprings. Considering μ the size of the parents population and λ to the number of descendants of the current generation, the strategies for manipulation of the population usually used are ([88]):

- (μ, λ) – parents are replaced by the offspring.
- $(\mu + \lambda)$ – offspring is added to the population, selecting μ individuals from $(\mu + \lambda)$.
- $(\mu + 1)$ – the parents generate a single offspring that replaces the worst parent of the population if it performs better.
- $(1 + 1)$ – a single parent produces a single offspring generated through mutation in an elitist manner.

Evolutionary Programming

EP was first applied to the evolution of finite state machines in the 1960's by L. Fogel [44]. In the following years EP research has strained only to be resurrected in the 1990's by the work of D. Fogel [18]. Fogel renewed the whole concept of EP but it became similar with ES in various aspects.

The role of Mutation is fairly equal to that seen in ES and the selection process is only slightly different, a form of Tournament Selection is used. One main difference is the absence of recombination, meaning that, all changes are applied via the mutation. Another difference is in the formation of individuals, that do not have a fixed structure or representation.

Genetic Algorithms

GA is the most widely known form of EAs. It was originally developed by John Holland and his students in the 1960's, altought their work extends until 1980's. The interest in GA is arguably related to the explosion of research in Artificial Neural Networks, since both are inspired in biological systems [88].

This approach is centered in the use of a genotype that is decoded and evaluated. The genotypes are, traditionally, simple data structures such as bit strings or real-value arrays that are evolved through recombination and mutation. An interesting aspect of this type of algorithm is identified by Holland [31] and DeJong [37] by describing GAs as a search process rather than an optimization process by focusing in the concept of competition. Recombination is the most emphasized operator in GAs and is traditionally done through a crossover operator which consists on exchanging parts of genotype between parents. In GAs, mutation is usually used with lower probability, in order to cause small disruptions at the gene level. Another key component of GAs is selection. Selection can be done by using various methods, the most usual being: roulette wheel, tournament and rank selection.

Genetic Programming

Unlike the other EAs, GP is not a parameter optimization method but rather fits in the scope automated programming. This approach was preceded by studies that can be seen as a form of genetic programming such as evolving finite state machines or classifier systems. However, GP introduced a major change in paradigm [41].

GP systems are often implemented in prefix notation, similarly to Lisp programs, i.e. the expression $((+(*-2x))4)$ represents the function $-2x+4$. Tree structures are typically used to represent syntactically correct expressions such as the one presented. Expressions are composed of functions and terminals that match, respectively, internal nodes and leaf nodes. The terminal set and the function set are human-defined and specify the components available in order to create individuals. In simple examples the function set may consist only of arithmetic operators and terminals of constants and external inputs. For more complex problems functions representing external tasks may be included.

The initialization of the population is usually done by the following methods: full or grow. The maximum tree size is usually specified. In the case of full initialization, nodes from the function set are used until the tree reaches

the maximum size and after that, only terminals can be used. In the grow initialization method nodes from both the function and terminal sets can be used until the maximum depth is reached. Another method for initialization is the ramped-half-and-half method where the full method is used to generate half of the individuals and the grow is applied to the other half.

Koza[41] defines the following genetic operators:

- reproduction – copy certain individuals into the new population;
- crossover – random pick a crossover point in each parent tree and swap the sub-trees associated;
- mutation – substitute a random sub-tree with a randomly generated one, sometimes implemented as a crossover between the individual and one randomly generated;
- architecture-altering – choose an operation from a one of the available sets and apply it on a given individual to generate an offspring;

Unlike GA, where mutation is applied after crossover, in traditional GP these operators are applied in exclusion, meaning that a selected individual will have a probability of indulging reproduction, crossover, mutation or architecture-altering operations. Genetic operators are usually applied until the offspring population is the same size as the parents' one.

Selection is done based on the fitness of each individual, with all methods of selection described previously being valid.

2.1.2 Evolutionary Computation for Image Generation

In the field of Evolutionary Computation for image generation, there is a need for a viable, scalable, expandable representation of the individuals, in this case, images [47].

Karl Sims in 1991, introduced the expression-based approach for evolving images [78]. This was done by using a GP algorithm and mathematical expressions as the genotype. An expression like $abs(max(x, 2 * \cos(y)) + \cos(y * \pi))$ can be represented as a tree graph structure, made up of mathematical functions and operators at internal nodes, and constants or variables at the leaves. When the expression represented by the tree is evaluated at each pixel in an image by plugging in the pixel's coordinates, the resulting value can be used to determine the color of a pixel. The resulting image is the phenotype. Images or forms can then be created and selected by using the usual recombination and mutation operators of a GP approach. There are still many different techniques for representing this genetic information. Choices about what functions to use, how to map values, originate different types of phenotypes, and can also influence the likelihood of finding interesting results.

The majority of these expression-based image generation systems use a reduced set of mathematical functions and often only local information for determining pixel color. There are systems that rely on specific techniques such as fractals, polar coordinate mappings, noise functions, texture applications, etc. Notice that specific additions to the function set or other system extensions can push system results in specific directions.

One pitfall of the majority of these image generation methods is the fact of being an interactive processes, that rely on the presence of an user to do the aesthetic judgement of the image in order to continue the process of evolution [24, 29, 22].

Other researchers have explored automatically evolving expressions using target images. This is the case of Ibrahim [35] which made some of the earliest attempts at replicating textures by using image analysis functions that evaluate rudimentary image features such as color, luminosity, and shape. Ross's initial work in this area with Wiens [89] sought to match simple test textures. Ross's more recent work attempts to generate expressions matching arbitrary artistic imagery.

Some hybrid systems using expression-based images were created, such as Baluja's [5] system that used neural networks previously trained with image examples to assign fitness and Machado's NEvAr system [59] that combines a evolutionary process, a classifier and a feature extractor to assign fitness. The work of Machado et al. has inspired the work of Atkins et al. [2] that uses a similar approach but use a different set of features.

In this section we made a brief overview of the state of the art in the areas that are foundations for this dissertation. The following sections cover concepts of other areas, which are necessary to understand the direction and focus of the developed work.

2.2 Image Classification

Considering the scope of this dissertation and the relation to the work presented in the chapter 3 this part of the state of the art is directed towards image classification, with an emphasis in aesthetics classification. A survey of classifier systems and a reflection about the used classification systems is presented.

Image classification is generally used to distinguish two or more classes, depending on the problem in hand. As a particular case, automatic image classification based on aesthetic criteria has multiple applications. It would allow image browsers to consider the aesthetic preferences of the user, enable online shops of artistic works, designs, or photographs to recommend works that are consistent with previous purchases, preferences or trends, etc. A system of this type could, for instance, be used to automatically classify artistic databases and then generate images of a given style or aesthetic current. This could be achieved by integrating the classifier as the evaluator of an evolutionary image generation system [5, 59]. In the referred works, Neural Networks were employed.

This is an area of continuous work and the rest of the section will present some examples of the classification tasks, the data sets involved and its classification performance.

Tong et al. [85] used a large set of low level features to distinguish between 12897 amateur photographs, taken by workers at Microsoft Asia, and a set of 16643 professional photographs, obtained from Corel Image Database and Microsoft Office Online. By using different classification methods, such as AdaBoost, Real-AdaBoost, Support Vector Machine (SVM), Bayesian and performing a 5-fold cross-validation on all 29540 images they have obtained a success rates above 91.1%.

Datta et al. [69] used high level ad-hoc features, based on color, texture and shape and Support Vector Machines to classify images gathered from a photography portal ("photo.net"). They considered two image categories: the most valued images (average aesthetic value ≥ 5.8, a total of 832 images) and least valued ones (≤ 4.2, a total of 760 images), according to the ratings given by the users of the portal. The system attains 70.12% classification accuracy.

Wang et al. [87] worked in high level features of the color histogram and tested its effectiveness in distinguishing between erotic and benign images. For that purpose SVM's and AdaBoost were used, attaining respectively, 85.32% and 87.91% performance.

In the work of Li an Chen [48], they have defined a group of high and low level features, such has color features, contrasts, image segmentation, contrast between shapes, in order to distinguish between high quality paintings and low quality. The classifier used was Ada-Boost. In their work, they have done a questionnaire to a group of people, in order to obtain subjective aesthetics scores from a dataset of 100 paintings (50 high and 50 low) and then confronted the results with their classification algorithm, which had a success rate of 74%.

In recent work, Wu et al. [92] have used low level features, inspired by the work of Datta [69]. The objective of the work consisted in a multi-class problem: distinguish between "beatiful", "divine" and "wonderful"; from "awful", "terrible" and "ugly". By using a dataset of 10800 images from flickr in a classification system that uses a variation of SVMs they attained an average success rate of 83.1% and 84.2%.

From the described state of the art work in image classification methods, it is suggested that the classifiers used, in spite of possible optimizations and different parameters choices, are mainly: Support Vector Machines [13], Neural Networks [21], Bayesian classifiers [45], AdaBoost [20].

2.3 Face Detection

In the field of Computer Vision, Face Detection has become a topic of great interest for the community. A Face Detection algorithm is a specific case of object-class detection, that attempts to identify the locations and sizes of human faces in an image. While this appears as a trivial task for humans, it is a challenging task for computers.

Many research projects and commercial products have demonstrated the capability for a computer to interact with human in a natural way by looking at people through cameras, listening to people through microphones, understanding these inputs, and reacting to people in a friendly manner. Furthermore, almost every photographic camera, every photo organizer application and even social networks has his own implementation of a face detection algorithm.

Face detection can be viewed as the stepping stone to all facial analysis algorithms, including face alignment, face modelling, face relighting, face recognition, face verification/authentication, head pose tracking, facial expression tracking/recognition, gender/age recognition, and many many more that are not listed here. The difficulty associated with face detection can be attributed to many factors about the images in evaluation. These include scale variations, location, orientation, pose, facial expression, lighting conditions and occlusions, among others.

In consequence of this demand and the general interest of the topic, there

Table 2.1: Features for face/object detection [95].

Feature type	**Features**
Haar-like features	Haar like features
	Rotated Haar-like features
	Rectangular features with structure
	On motion filtered image
Pixel Based features	Pixel pairs
	Control point set
Binarized features	Modified census transform
	Local binary pattern (LBP) features
	locally assembled binary feature
Generic linear features	Anisotropic Gaussian filters
	Local non-negative matrix factorization
	Boosting Generic linear features
	Recursive nonparametric discriminant analysis
Statistics-based features	Edge orientation histograms
	Spectral histogram
	Spatial histogram
	Histogram of oriented gradients and LBP
	Region covariance
Composite features	Joint Haar-like features
	Sparse feature set
Shape features	Boundary/contour fragments
	Edgelet
	Shapelet

have been numerous approaches to face detection [43, 30]. Before the year 2000, Yang et al. [43] have divided the various solutions into four distinct categories: knowledge-based methods, feature invariant approaches, template matching methods, and appearance-based methods. Knowledge-based methods determine the detection of a face based purely on human experience and knowledge; feature invariant approaches aim to find face structure features; template matching methods use pre-stored face templates to judge if an image is, in fact, a face; appearance-based methods learn face models from a set of representative training face images to perform detection. In general, the most successful were the appearance-based methods that showed superior performance when confronted with other approaches.

Although the field of face detection has made significant progress in the past decade, the work by Viola and Jones [86] has made face detection feasible in real world applications such as digital cameras and photo organization software. Their work on boosting-based face detection schemes, has become the de-facto standard of face detection in real-world applications since then. In a recent survey of Zhang et al. [95] it is referred that if one were asked to name a single face detection algorithm that has the most impact in the 2000's, it will most likely be the work made by Viola and Jones.

Nowadays there are three key issues that need the attention in a face detection system: what features to extract; which learning algorithm to apply; how to build or chose a dataset for training and testing purposes.

In table 2.1 there is a summary of the features usually extracted from images in order to perform the detection. In what concerns the type of classifier and learning algorithm to use, a considerable number of variations to the approach of Viola and Jones have been made and several other approaches exist. Table 2.2 summarizes algorithms that rely on boosting while table 2.3 displays other

Table 2.2: Schemes to address challenges in boosting learning [95].

Challenges	**Proposed slutions**
General boosting schemes	Adaboost
	Realboost
	Gentleboost
	Floatboost
Reuse previous node's results	Boosting chain
	Nested cascade
Introduce asymmetry	Asymmetric boosting
	Linear asymmetric classifier
Set intermediate thresholdsduring training	Fixed node performance
	WaldBoost
	Based on validation data
	Exponental curve
Set intermediate thresholds after training	Greedy search
	Soft cascade
	Multiple instance pruning
Speed up training	Greedy search in feature space
	Random feature subset
	Forward feature selection
	Use feature statistics
Speed up testing	Reduce number of weak classifiers
	Feature centric evaluation
	Caching/selective attention
Multiview face detection	Parallel cascade
	Pyramid structure
	Decision tree
	Vector valued boosting
Learn without subcategory labels	Cluster and then train
	Exemplar-based learning
	Probabilistic boosting tree
	Cluster with selected features
	Multiple category boosting

noticeable approaches for face detection. These include the traditional neural networks, support vector machines, bayesian classifiers, regression and templates obtained through constrained optimization.

These facts can lead to conclude that there is already a lot of work done in face detection and that investigation has been continuously evolving until today. The future direction, consists in further improvement of the learning algorithm and features. For instance, the Haar features used in the work by Viola and Jones are very simple and effective for frontal face detection cases, but they seem to fail when detecting faces at arbitrary poses.

The modern face detectors are mostly appearance-based methods, which means that they need training data to learn the to perform the classification task. Creating an adequate dataset is a complex and time consuming task, involving the collection and classification, often by hand, of large amounts of training data, which certainly demands more research. Schemes such as multiple instance learning boosting and multiple category boosting are helpful in reducing the accuracy needed for the labeled data, though ideally one would like to leverage unlabeled data to facilitate learning. An unsupervised or semi-supervised learning schemes would be possible solution for this issue. Additionally, promoting the representativeness and completeness of the dataset is, to a large extent, an open research question.

Table 2.3: Other schemes for face detection [95].

General Approach	**Representative Works**
Template matching	Antiface
Bayesian	Bayesian discriminating features
SVM–speed up	Reduced set vectors and approximation
	Resolution based SVM cascade
SVM– multiview face detection	SVR based pose estimator
	SVR fusion of multiple SVM's
	Cascade and bagging
	Local and global kernels
Neural networks	Constrained generative model
	Convolutional neural network
Part-based approaches	Wavelet localized parts
	SVM component detectors adaptively trained
	Overlapping part detectors

Chapter 3

Proof of Concept and Experimentation

In this chapter the preliminary research and experiments that inspired and motivated the current dissertation, making us abandon the original proposal, is presented. The purpose of this work was to generate images that resembled human faces by means of an EC engine. It involves EC, CV and ML techniques. The difficulties encountered, and the unexpected results of the task, led to interesting insights and observations, creating the desire to address the identified issues.

The approach followed here is informed by previous research where a classifier is used to guide evolution, namely the work of Machado et al.[59]. While in Machado et al. the employed classifier was developed by the team, in this case we use a pre-existent classifier, developed by other researchers. The original motivation for using a classifier developed by others was to highlight the generality of the proposed technique by demonstrating that it did not require a special-purpose classifier specifically tailored for the task.

In the remainder of the Chapter we describe this preliminary work, presenting results and drawing conclusions.

3.1 Evolving Faces

As previously stated, the goal was to evolve images resembling human faces. To attain this goal an evolutionary image generation engine and a face recognition algorithm are required.

The EC engine used was, NEvAr, developed by Machado et al.[54], an Evolutionary Art tool, inspired by the work of Karl Sims[78] and Richard Dawkins[14]. NEvAr employs Genetic Programming. The genotype of an individual is a tree, built with a lexicon of functions and terminals, while the phenotype is an image, resulting from the interpretation of the genotype over the pixel value coordinates. In the initial versions of NEvAr, the evolutionary engine followed this steps of execution:

1. Random initialization of the population.
2. User assisted fitness assignment.

3. Generate a new population with the application of the chosen genetic operators, where the population individuals with higher fitness values have higher probabilities of being selected.

4. Repeat 2 until some stop criterion is met.

Later the system expanded to automatic methods of fitness assignment (see, e.g, [54, 59]) using both heuristics and ML approaches to assign fitness. NEvAr was chosen because of its potential for image generatio, for its well-established role in the research community [58, 54, 60, 55] and for being easy to adapt for the problem in hand. It is important to notice that, from a theoretical standpoint, NEvAr has the potential to generate any image. In other words, it is possible to represent any image using the representation scheme employed in NEvAr and, as such, any image is theoretical evolvable [54].

The face detection algorithm used was the one from Viola et al.[86] to detect frontal faces. The code and executables are included in the OpenCV API. This algorithm was chosen because of its state of the art relevance, for its fast classification and by the simple methodology behind the method. This algorithm uses a set of small features in combination with a variant of the Ada-boost [20], in order to attain extremely efficient classifiers. To accomplish this objective, the approach makes use of a cascade of small classifiers and use Haar features [67], i.e., rectangular features, that are calculated through the integral image method. An example of the integral image method is presented in figure 3.1.

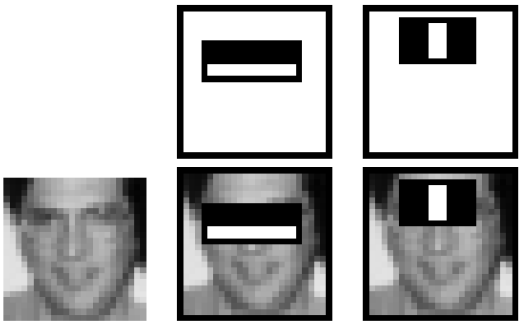


Figure 3.1: Haar features [86].

In this case, two-rectangular features were used. By subtracting the pixels of the black zone with the white zone we obtain the feature result. If this result is superior to a given threshold, then the tested feature is present on the image. Initially, only vertical and horizontal were used, but the work of Lienhart [51] introduced several extensions to the used features (see figure 3.2).

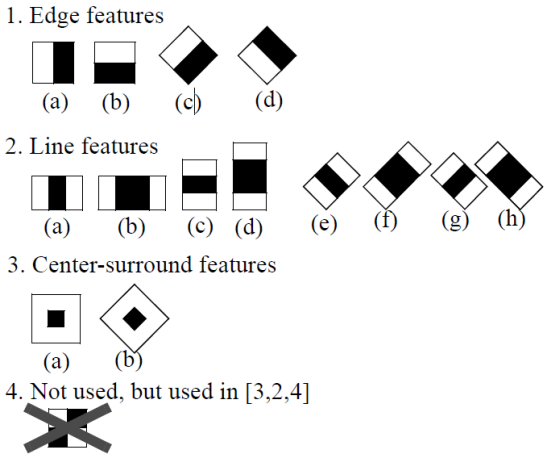


Figure 3.2: Extended Haar features [51].

The face detection process can be summarized in these steps (parameters from Viola et al. work [86]):

1. Define a window of size w (20×20).
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 has the size 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. The cascade has a group of stage classifiers, as represented in figure 3.3. Each stage is constituted at its lower level for a group of Haar features. Apply each feature of each stage to the sub-window. If the resulting value is lower than the stage threshold the sub-window does not have a face and the search terminates for the sub-window. If it is higher continue to next stage. If all stages are passed, the sub-window has a face.
6. Apply the scale factor s to the window size w and repeat 5 until window size exceeds the image in at least in one dimension.

In order to conduct this experiment, three classifiers were used. These were obtained from Lienhart's[51] website[1] and will be named C1, which uses the "alt.xml" file; C2 ("alt2.xml"); C3('default.xml").

Figure 3.4 presents an overview of our system, which is composed of two main modules, the GP engine and the face detection algorithm (FD).

[1]Haar Cascades [51]- http://alereimondo.no-ip.org/OpenCV/34

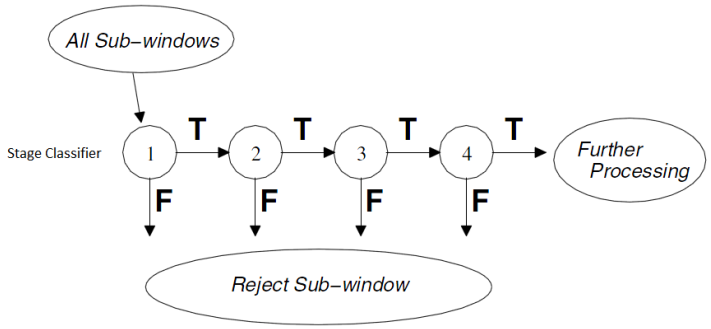


Figure 3.3: Cascade of classifiers with N stages [51].

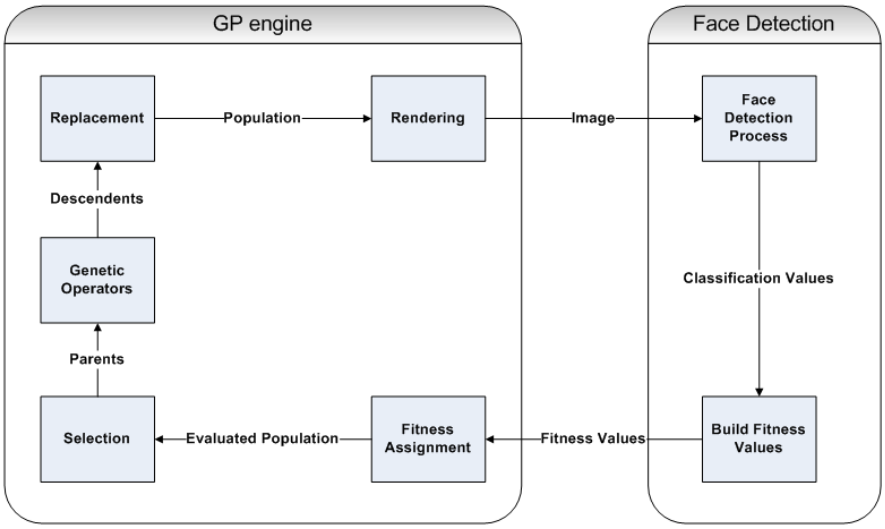


Figure 3.4: Overview of the system.

Execution proceeds as follows:

1. Random initialization of the population;
2. Rendering of the individuals, i.e., genotype-phenotype mapping;
3. Apply the face detection (FD) to each individual;
4. Use classification values from FD to build the fitness (se formula 3.1);
5. Select progenitors; apply genetic operators, creating descendants; Apply he replacement operator resulting in a new population;
6. Repeat 2 until some stop criterion is met.

The process of fitness assignment is crucial from a evolutionary point of view, and therefore it holds a lot of importance for the success of the described system.

In our case, we wish to use the outputs of the face detection algorithm are used to assign fitness. However, the FD algorithms employed produce a boolean output, which is inappropriate to guide evolution. A binary function gives no information of how close an individual is to being a valid solution to the problem and, as such, with a binary function the EA would be as effective as random search.

Ideally, the fitness function should provide a smooth fitness landscape that promotes evolution. It is, therefore, necessary to extract additional information from the FD in order to build a suitable fitness function.

In several informal experiments, we focused on developing an appropriate fitness function by analyzing the results of several runs, trial and error, incremental improvements and refinements, etc. This implied studying and changing the source code of the classifiers in order to collect the necessary information. However, these changes have no influence regarding what is and is not classified as a face. Eventually we settled with the following fitness function:

$$fitness(x) = \sum_{i}^{countstages_x} beststagesum_x(i) * i + countstages_x * 10 \qquad (3.1)$$

The variables $countstages_x$ and $beststagesum_x(i)$ are extracted from the FD algorithm. Variable $countstages_x$,holds the number of stages the image has successfully passed in the cascade of classifiers. The rationale is the following, an image that passes several stages is likely to be closer to being recognized as having a face than one that passes very few stages. In other words, passing several stages is a pre-condition to being identified as a face image. Variable $beststagesum_x(i)$ holds the maximum difference between the threshold and the integral image ofthe i^{th} stage. The rationale is the following, we consider that images that are clearly above the thresholds necessary to pass each stage should be valued over ones that are just above it.

Using this fitness function we conducted several experiments. The experimental setup comprised the usage of one classifier for fitness assignment, while observing the behavior of the others. For each classifier, 10 different runs were created and, for each run, the evolutionary algorithm performed 100 iterations. Due to the experimental nature of this work, this setup was thought in order to perform fast and relevant tests.

Figures 3.5, 3.6 and 3.7 summarize the results attained using this approach in terms of mean fitness and maximum fitness per run. In each of the charts, the bold lines indicate the classifier used to guide fitness while the other lines pertain classifiers that did not influence the evolutionary run. Note that the data is normalized based on the mean maximum achieved in each classifier's test. The fitness values attained using different classifiers cannot be directly compared.

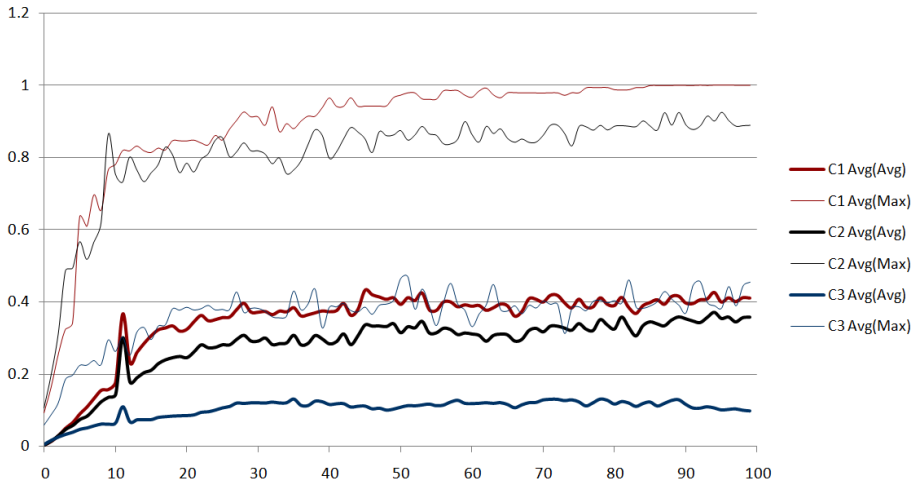


Figure 3.5: Evolution of the average and maximum fitness when using the C1 classifier to assign fitness. Results are averages of 10 runs.

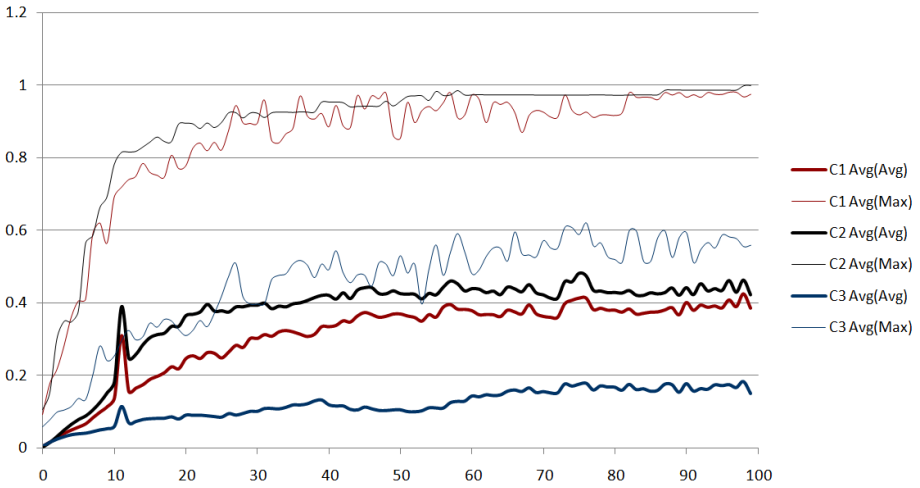


Figure 3.6: Evolution of the average and maximum fitness when using the C2 classifier to assign fitness. Results are averages of 10 runs.

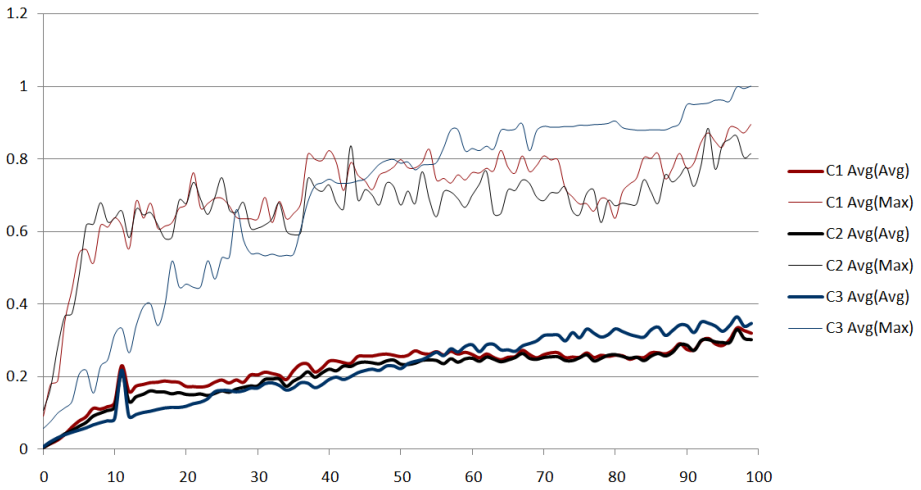


Figure 3.7: Evolution of the average and maximum fitness when using the C3 classifier to assign fitness. Results are averages of 10 runs.

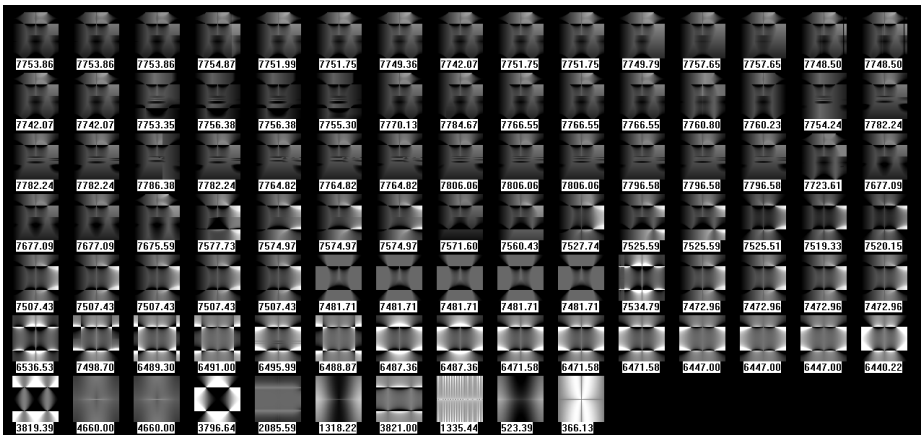


Figure 3.8: Fittest individual per generation from last to first generation in reading roder.

The results indicate interesting aspects of the experimental process. In terms of maximum fitness, it suggests that when c1 fitness increases, c2 seems to increase almost at the same rate. They also indicate that c3 achieves the lowest performance, except when it is guiding the evolutionary process. The c2 classifier seems to promote the c3 performance more than c1 does. These experimental results also suggests that, similar to the findings of Viola and Jones [86], c3 is the more robust of the classifiers.

Figure 3.8 shows the evolution of the best individual, per generation, during the course of the run that we consider to be best in terms of evolving images that actually resemble faces. This is, obviously a subjective evaluation and this

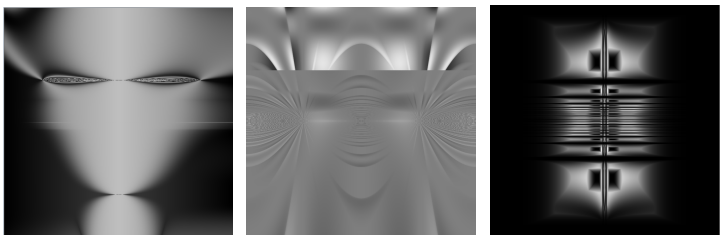
image is included only for illustrating evolution.

This run is atypical since it actually evolved mages that, in our subjective opinion, actually resemble faces. The typical results is the following: the runs evolve images that the classifiers identifies as faces but that, a human would probably never classify as possessing a face. In other words, in most runs what the EA found were images that are false positives. It is interesting to notice that, most of this false positives are misclassified not only by the classifier guiding evolution but also by the other classifiers considered.

In figures 3.9, 3.10 and 3.11 we present, for each classifier, six images evolved using it to assign fitness. The images are labeled with the classification assigned by the classifier. These images were chosen to illustrate the strengths and weakens of the classifiers,

Looking, for instance, at 3.9a and 3.9f it is easy to understand, and even agree, with the classification made by c1. However, it is surprising to see image 3.9c and 3.9d classified as faces. The same applies to 3.10a; 3.10a may be evocative of a human face, however for images 3.10b to 3.10d we find it difficult to believe a human would classify them as such. Surprisingly, in our eyes, 3.10e is more evocative of a human face than the previous ones. Likewise, and although it is always subjective to say it, it appears reasonable to state that a human would not identify images 3.11a, 3.11b and 3.11c as being human faces.

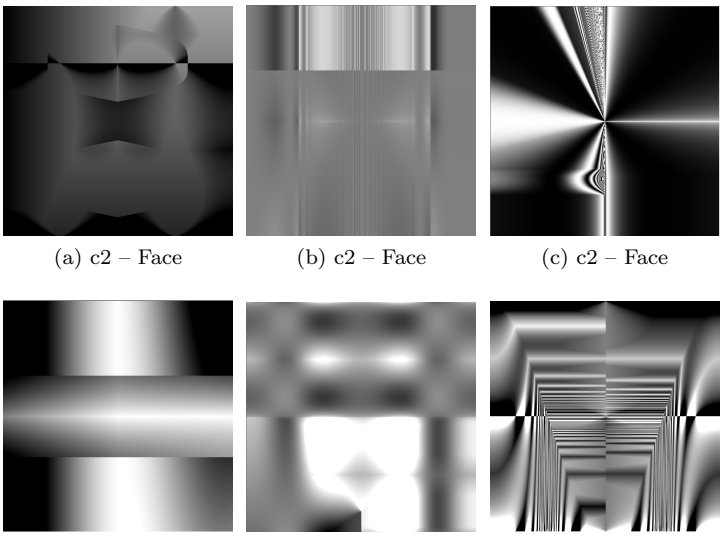
In the next section the conclusions of this work and the insights about this approach, are described.



(a) c1 – Face (b) c1 – Face (c) c1 – Face

(d) c1 – Face (e) c1 – Face (f) c1 – Not Face

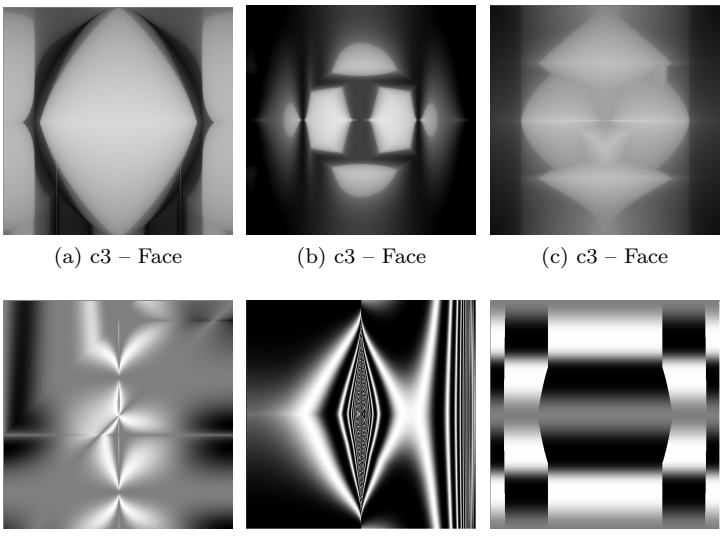
Figure 3.9: Examples of individuals evolved using c1 and the classification assigned by the FD algorithm.



(a) c2 – Face (b) c2 – Face (c) c2 – Face

(d) c2 – Face (e) c2 – Not face (f) c2 – Not face

Figure 3.10: Examples of individuals evolved using c2 and the classification assigned by the FD algorithm.



(a) c3 – Face (b) c3 – Face (c) c3 – Face

(d) c3 – Face (e) c3 – Not face (f) c3 – Not face

Figure 3.11: Examples of individuals evolved using c3 and the classification assigned by the FD algorithm.

3.2 Conclusions and Insights

The preliminary experiments conducted led to some relevant conclusions. The presented approach indicates that a ML method can guide and EC algorithm. As a proof of concept, it was possible for a EC image generator to create images that were classified as faces by a FD. Additionally, the proposed formula used to assign fitness showed to be valid, providing a fitness landscape that enables the EA algorithm to maximize fitness.

These results also showed that most of the evolved images classified as faces by the FD would probably not be classified as faces by a human observer. It is important to notice that this tendency for false positives does not reflect a weakness of the proposed fitness formula. For better and for worst, fitness can be maximized by generating false positives, but this is not a deficiency of the proposed formula, it is a weakness of the used classifiers.

In a nutshell, the tendency of the EA to evolve false positives highlights both the shortcomings of the FD algorithm and the ability of the EA to exploit these shortcomings.

From the attempts to solve these limitations, a new idea has emerged. The EA is in fact synthesizing examples, that may, arguably, be used to assess and improve the classifier performance.

In Machado et al. work [59] a related approach was used in order to construct an Artificial Artist. The results indicated that the approach is capable of improving the classifiers based on Neural Networks. The preliminary experiments presented herein indicate that it is possible to adapt an Ada-boost binary classifier to guide evolution and that, in this case, the EA also appears to have a tendency to use "shortcomings" to maximize fitness. Recent work, were a Support Vector Machine based classifiers [52] was used to assign fitness in the context of an evolutionary image generation engine, indicates that the overall concept may also be applicable in that case. These results leads to conclude that it may prove valuable to develop a framework for assessing and improving classifier performance by using EAs to synthesize training and test instances. As far as we know this proposal is novel. With the exception of the work of [59] EA were never used for this purpose and the scope of [59] was smaller and the goals different, namely attaining style variation instead of improvement and assessment of classifier performance.

Chapter 4

Framework

This chapter makes a brief outline of the framework we propose for the improvement and assessment of classifier performance. This framework relies on the use of EC for the the synthesis of new examples for training and testing purposes, which constitutes, to the best of our knowledge a novel approach.

We focus on CV tasks and hence on the generation of new testing and training images. Nevertheless, the presented framework is not specific to CV and can be applied in other domains.

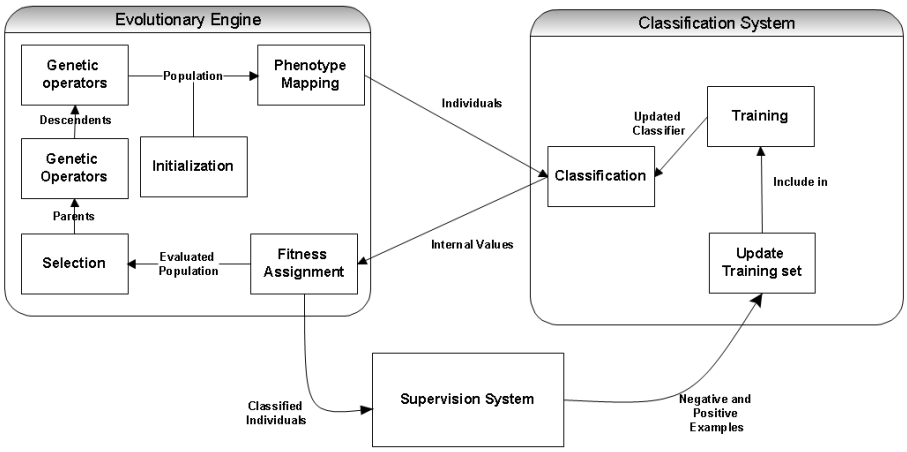


Figure 4.1: Overview of the framework.

Figure 4.1 presents an overview of the proposed framework. Which is composed of three main modules: Evolutionary Engine, Classification System and Supervisor System. The role of the evolutionary engine is to evolve new examples; the role of the Classifier is to classify those examples; and the role of the supervisor, is to determine which examples are correctly and incorrectly classified.

The overall methodology shares many similarities with the boosting approach proposed by Machado et al.[59]. Considering that a Classifier System, CS, exists and that the system was trained with a given *dataset*, the approach

consists in the the following steps:

1. A new evolutionary run is started – a initial random population is created, and evolution proceeds as regularly, CS is used to classify each individual and the returned values (which may include internal values of the classifier as we have seen in the previous Chapter) are employed to assign fitness.

2. The EC run stops when a given termination criterion is met (e.g. after a predetermined number of populations, upon reaching a solution, upon finding n examples of a given class or classes, etc.);

3. The supervision system determines which individuals have been misclassified by the CS, these individuals are added to the current *dataset*.

4. The CS system is retrained using the expanded *dataset*.

5. Unless a termination criteria is met, a new boosting iteration is started by repeating the process from step 1.

The entire approach relies on promoting a competition between the evolutionary engine and the classifier system. In each iteration the evolutionary engine must evolve images that are misclassified by the CS, otherwise no progress is achieved.

Implicitly, this implies that the evolutionary engine should be more likely to produce images of a particular class. For instance, in normal circumstances NEvAr is unlikely to produce images resembling faces. By assigning fitness using a face detection classifier, like we did in the previous Chapter, and valuing images that are classified as faces, we were able to evolve several misclassified images. These misclassified images are the ones that can be useful for improving the performance of the classifier.

To assign fitness it is necessary to develop a fitness function that can successfully guide evolution; This can imply having access to internal variables and calculations of the classifier.

The supervision system may be an human observer or a different classifier system. Using a human classifier can be time-consuming and humans are prone to errors. Nevertheless, in some scenarios, human supervision is required since no competitive classifiers exist. Moreover, if one is only interested in the addition of misclassified examples of a given class (e.g. images wrongly classified as faces) the number of examples that the human needs to inspect is far less than the total number of individuals, which makes the task tractable. Using an external classifier implies that the approach is, at least partially, limited by the performance of the supervisor system. However, it is important notice that the weaknesses of the supervisor system cannot be directly exploited by the evolutionary process, as such this may be a case were the whole is greater than the some of its parts. Like in the human supervisor scenario, the number of individuals that need to be classified by the supervisor may be a lot smaller than the number of individuals, which opens the door to the use of supervision classifiers that are more complex than the ones being used to assign fitness and to the use of sets of classifiers in supervision.

Finally, it is necessary to determine an appropriate termination criterion. This depends, mainly, on the task at hand and on the existent computational resources.

In the Following chapters we describe the application of this generic framework to two different image classification problems, presenting and analysing the experimental results attained and drawing conclusions.

Chapter 5

Style-Based Image Classification

The work presented in this chapter is inspired by the work of Machado et al.[59]. The similarities and differences, the experimental setup employed, the results attained are described and analyzed. The chapter starts with a brief introduction to the work, explaining the work of Machado et al. and presenting an outline of the experiments (section 5.1). In section 5.2 we instantiate the framework proposed in the previous chapter to the current problem. The experimental setup is described in section section 5.3 and the experimental results presented and analyzed in section 5.4. Finally, the results attained in several validation experiments are presented, section 5.5, and conclusions are drawn (section 5.6).

5.1 Introduction

The work of Machado et al. describes the creation of an Artificial Artist, capable of aesthetic and/or artistic judgements. Its architecture comprised two modules: the Creator and the Artificial Art Critic (AAC). The Creator module consist in an EC engine and the AAC is composed by a Feature Extractor (FE) and an Artificial Neural Network (ANN). The system has two characteristics [59]:

1. The use of an ANN to distinguish between internal images (generated by the EC engine) and a selected set of external images (e.g., famous paintings, artworks of a given style, landscape photographs, portraits, etc.).

2. The iterative execution of the following steps:

   - The EC engine attempts to find images that are classified as external ones; the ANN's output is constructed in function of the individual's fitness.
   - In each EC run, the created images are added to the training set of the ANN as instances of internal images.
   - The ANN is trained to distinguish between the sets.

The experiments results showed that the AAC was capable of discriminating between the two sets and guide the EC algorithm. It has also indicated that the EC algorithm was able to find internal examples that were classified as external.

The purpose of remaking this experiment is to further explore the above conclusions and to understand if it is possible to assess and improve a classifier performance, through the use of the framework presented in the previous chapter, which exploits the vulnerability of the classifier to false positives, generated by an EC algorithm, and the evolution of the classifier performance when using these examples in a new training stage.

Although the overall approach is similar to the one presented by Machado et al. [59] there are important differences both in terms of objectives and implementation. The goal of Machado et al. was to promote stylistic change from one evolutionary run to the other, here, although stylistic change may occur, we are primarily interested in improving the performance of the classifier. In terms of architecture, the main difference is the existence of a supervisor module, which was not present in the work of Machado et al. The differences in objectives implied several changes at the implementation level. The most notable ones are: (i) The FE extractor was augmented and some of the features re-implemented; (ii) Several classifier feature selection methods were employed; (iii) New, and significantly larger, initial datasets were constructed; (iv) Several dataset update strategies are considered.

These and other important issues will be covered further in the following sections were we describe the approach, section 5.2, and the experimental setup, section 5.3.

5.2 The Approach

The boosting algorithm employed is an instantiation of the framework presented earlier. It consists of the following steps:

1. An *external* set of images is provided to the algorithm;
2. The EC engine creates, randomly, an *internal* image set.
3. A Classifier System (CS) is built and trained to distinguish between *internal* and *external* image sets.
4. A new EC run is started. The output of the CS is used to assign fitness.
5. The EC run stops when a termination criterion is met (e.g., a pre-established number of generations, attaining a fitness value).
6. The set of *internal* images is updated by the Supervisor which determines which images should be added.
7. The process is repeated from step 3 until the boosting criterion is met.

As previously mentioned, the *external* image set is composed of images that were not generated by the EC engine and that, in this case, are considered of high aesthetic value, the *internal* image set contains images generated by the EC engine. In this case we have two clearly defined classes, either the image

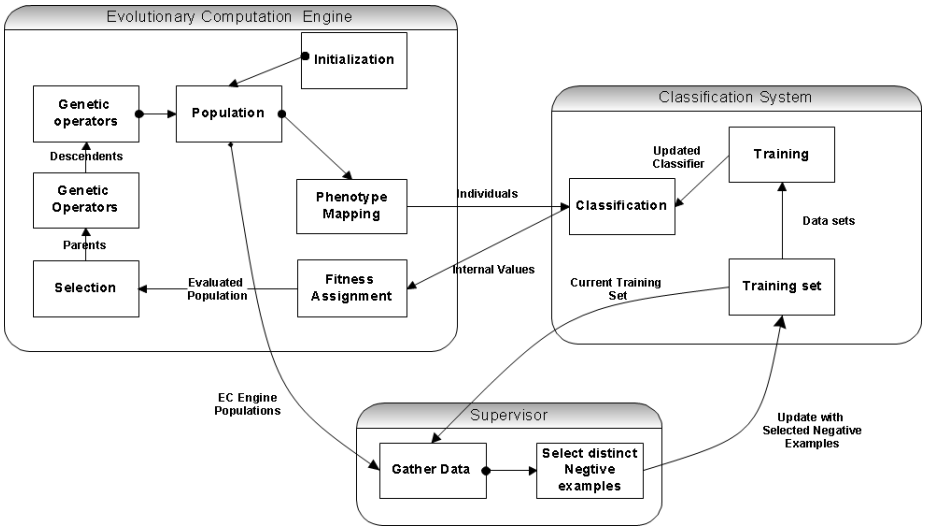


Figure 5.1: System overview.

was generated by the EC engine or it wasn't, so there is no subjectivity in the classification.

However, since NEvAr has the potential to create any image, the image sets may eventually overlap. Additionally, the external image set is composed of paintings and, as such, in a best case scenario it is representative of paintings. Thus if the classifier is confronted with an image that was not generated by the EC engine but that also is not a painting – e.g. the photo of a car, a chart – the correct classification for this image is somewhat subjective: it is not a painting but it was also not created by th EC engine.

The overview of the system is shown in figure 5.1.

The EC engine used is the same has NEvAr's [54]. As previously mentioned, NEvAr is an expression based Evolutionary Algorithm system that allows the evolution of populations of images. It is a system that employs GP.The genotypes are trees composed from a lexicon of functions and terminals. The functions include mathematical and logical operations; the terminals are composed by two variables, x and y, and random constant values. The phenotype is the image and it is generated by evaluating the genotype along the x and y coordinate of the image. Genetic operators, such as recombination and mutation, are performed at the genotype level.

The Artificial Art Critic suffered several modifications. It is still composed by an FE and an Evaluator but is also connected to a Supervisor, a new part of the system. The Feature Extractor was rebuilt and greatly modified from the one used in [59]. Its purpose is to analyze the image, collecting characteristics that maybe useful for classification. The features extracted are, mainly, related with the estimation of image complexity. A full description of the FE can be found in Appendix A.

Figure 5.2 presents a grid of existent features distinguishing between the features used in [59], features that were re-implemented during this dissertation, and features that were developed in the scope of this dissertation.

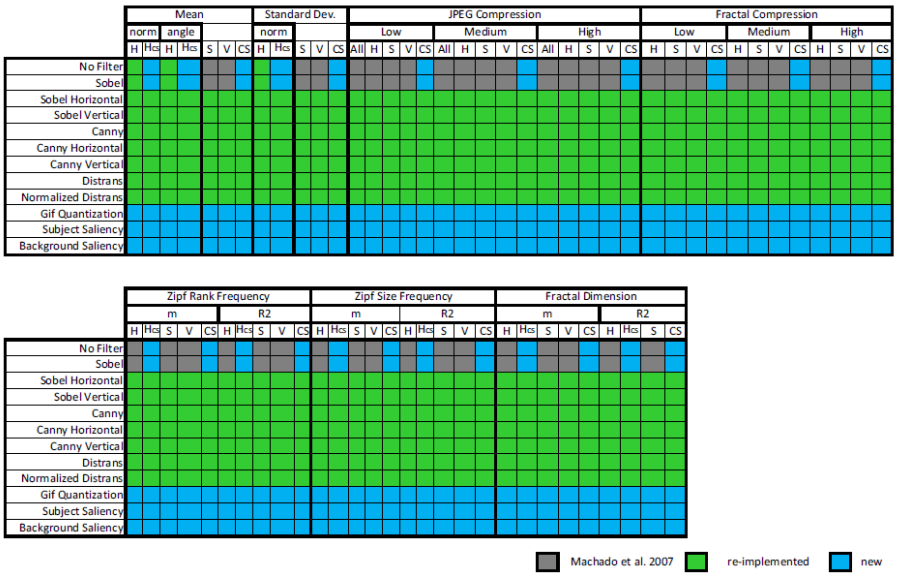


Figure 5.2: Features available for each partition.

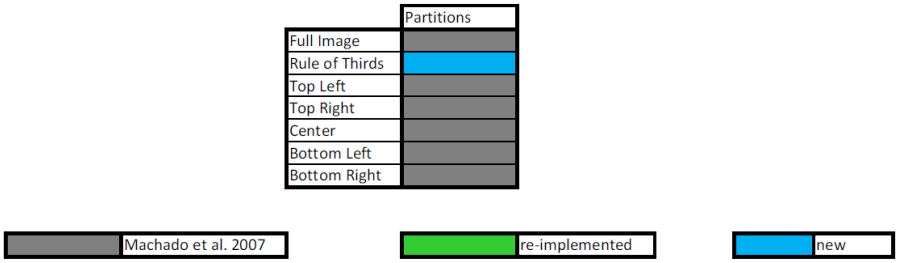


Figure 5.3: Image partitions.

The feature extraction process allows the partition of the image and the calculation of the features for each of the partitions (see Appendix A). The existent partitioning schemes and the novel one are presented summarized in figure 5.3. In the current experiments we did not employ image partitioning, the features are only calculated for the entire image. Even in this scenario the FE collects a total of 804 feature values which are then used by the CS for classification.

The CS is composed by an ANN and a feature selection (FS) method.

The ANN is a feed-forward network, with one hidden layer and one output neuron. It is trained with standard backpropagation. The classifier was built using WEKA's[1] FastNeuralNetwork. WEKA is a workbench for machine learning with a significant number of algorithms and tools available, used in a vast number of experiments [32, 26, 96, 83]. The choice of an ANN based classifier

[1]WEKA 3: Data Mining Software in Java - http://www.cs.waikato.ac.nz/ml/weka/

is justified by the success of this approach in [59].

A FS method is typically composed by an *evaluation* criteria and a *search* method. In what concerns *evaluation* criteria there are three main choices: *Filter* – selects a set of features based on the predictability of the available features, which is usually assessed by a statistical method (e.g., chi-squared, ANOVA, correlation); *Wrapper* – selects a set of features to train and test a classifier, and scores the group of features based on the classifier performance; *Hybrid* – uses a combination of the methods filter and wrapper methods. The *search* methods can be adapted from generic search algorithm and they are grouped in either *optimal* or *sub-optimal*. Among the optimal search methods exhaustive search and branch and bound can be considered. The first one has a complexity of $2^{\#features}$, which, in most of the cases, makes it unfeasible to apply. Branch and bound optimality requires that increasing the number of features never degrades performance. In practice this happens often. *Sub-optimal* search methods include deterministic and stochastic approaches.

From the above description it becomes evident that a vast number of combinations of evaluation and search methods exists. We chose to use a *filter evaluation* method with a sub-optimal *sequential search* algorithm. The necessity of adopting a cost effective approach, due to the lack of computational resources, made us adopt *filter* evaluation and *sub-optimal* search. The choice of deterministic search approach makes the analysis of the results less complex. Considering these issues, we employ CfsSubsetEval [27] for evaluation and best first search. The CfsSubsetEval evaluates the worth of a subset of features, by statistically process each feature, in terms of information redundancy and correlation with the class. Resulting subsets of features tend to be highly correlated with the class and low intercorrelated. The best first algorithm, defines the CsfSubsetEval as its heuristic function and scores nodes near its starting point, then it expands to the available node of features with highest score. The search stops when a pre-determined number of non improving nodes are encountered. Both of these algorithms are provided by the WEKA tool and were integrated in the system. FS is used in step 3 of the algorithm before the training phase of the classifier.

Another new part of the system is the Supervisor. It has the purpose of choosing the individuals that will added to the image datasets. In this particular experiment, it picks individuals from the EC run that are not present in the *internal* set and sequentially substitutes the existing ones. The Supervisor compares individual by comparing the feature values. This means that the Supervisor modifies the internal set, by eliminating the old examples iteratively and including unique individuals in their place, generated during the EC run, until the boosting process finishes.

The integration of the CS and EC systems involves a series of steps: (i) rendering the individuals; (ii) applying the FE to extract the features; (iii) filtering the resulting features using the feature subset previously calculated by the FS selection method; (iv) feeding the resulting feature subset to the ANN; (iv) calculating the ANN output for each image, i.e., the classification; (v) mapping the outputs of the ANN to fitness values.

Like in the previous experiment, there is a tolerance threshold to avoid binary output predictions and consequently a non suitable fitness landscape. This means that during the backpropagation of the error, if the difference between the output of the network and the desired output is bellow the maximum tolerated

Table 5.1: NEvAr's parameters.

Parameter	Setting
Population Size	50
Number of generations	50
Crossover probability	0.8
Mutation operators	sub-tree swap, sub-tree replacement, node insertion, node deletion, node mutation
Initialization method	ramped half-and-half
Initial maximum depth	5
Mutation max tree depth	3
Function set	+, -, *, /, min, max, abs, sin, cos, if, pow, mdist, warp, sqrt, sign, neg
Terminal set	X, Y, scalar and vector random constants

Table 5.2: FE's parameters.

Parameter	Setting
fractal compresion	2 , 4 , 2 , 5 , 3 , 6
jpeg quality (low, normal, high)	20 , 40 , 60
filters	ALL
metrics	ALL
channels	ALL
#colors from quantization	16
subject salience #kernels	30
subject salience neighbouring window	3

threshold, then the error is propagated back as zero (no error). As it will be shown in section 5.4 this enables us to guide the EC engine to promising areas of the search space.

5.3 Experimental Setup

This section describes the experimental settings, regarding the EC system, the FE, the CS composed by an ANN and a FS method and the datasets in use. In the end of the section the experiments conducted are described.

NEvAr's settings are presented in table 5.1 and based in previous works conducted by Machado et al. [54, 59]. The individuals are rendered to a size of 128×128. Since the population size is 50 and the number of generations is 50 a total of 2500 individuals is generated in each evolutionary run. The random number seed in all stochastic methods is always restarted, meaning, for instance, that the initial population of each EC run is always the same in all iterations. The initial population can be viewed in B.1.

The FE processes all images in a format 128×128 and 8 bits per channel (pixel values between 0 and 255), so each input image will be converted to that same size. The configuration of the FE is described in table 5.2 and further information on the FE is presented in section A.

The parameters of the ANN and FS can are described in table 5.3.

The ANN is composed by 15 neurons in the hidden layer, was trained during 500 iterations, with a momentum of 0.2 which is applied to the weights during the update, a learning rate of 0.3, the input attributes are normalized between 0 and 1 and the tolerance threshold is of 0.3. The percentage of negatives and positives is in one to one ratio and the training is made with 90% of the input

Table 5.3: CS's parameters.

Parameter	Setting
% train	0.9
% test	0.1
% positives	0.5
% negatives	0.5
#neurons hidden layers	15
learning rate	0.3
momentum	0.2
epochs	500
normalize attributes	yes
tolerance	0.3
FS evaluator	CfsSubsetEval
FS search Method	BestFirst
FS #N max nodes	10

data sets. Regarding the FS, the chosen maximum number of nodes without improvement was 10.

The *external* data set holds 25000 paintings from different authors, styles and periods (figure 5.4). Among others, it includes paintings of: Michelangelo, Picasso, Monet, Gaugin, Dalí, Cézanne, Da Vinci, Manet, Matisse, Chagall; among others. The *internal* data set is a randomly generation population of 25000 individuals (figure 5.5).

The ANN parameters imply that hat 45000 images will be used for training and 5000 for testing putposes, 22500 *internal* and 22500 *external* images for training and 2250 of each for testing. The boosting algorithm stops when 22500 of the internal image set is replaced by novel images.

The desired output for external images is 1 while the desired output for internal images is 0. The fitness of an individual is equal to the value return by the output neuron of the classifier.

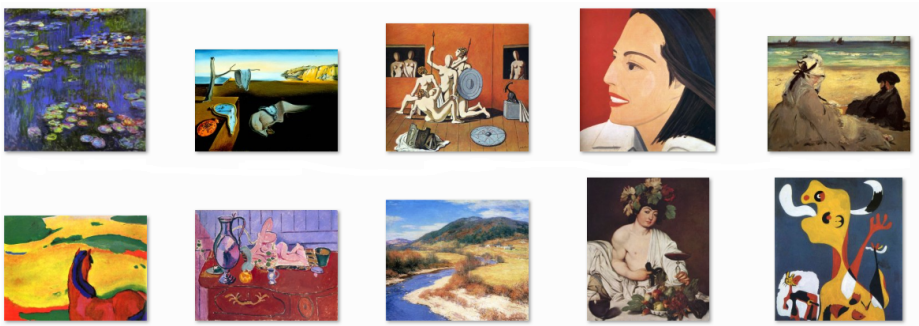


Figure 5.4: Samples of external data set.

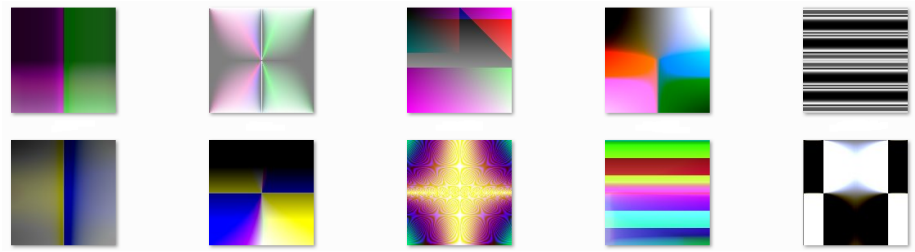


Figure 5.5: Samples of internal data set.

We conducted three independent experiments with different feature selection methods:

1. FULL – No feature selection method is used and, therefore, the CS uses the entire set of features;
2. FW – uses CfsSubsetEval with forward selection best first search;
3. BW – uses CfsSubsetEval with backward selection best first search;

When forward selection is used the selection algorithm starts with an empty set of features and it incrementally adds features until a termination criterion is met. Backward selection starts with the full set and removes features until the criterion is reached. From here on, we will use the terms FULL, FW, BW to refer to the each of the experiments.

5.4 Experimental Results

In this section the experimental results are presented and discussed. First we focus on the initial iteration of each experiment. Subsequently, we analyze each experiment regarding performance, features used and false positives across iterations. Then we focus on the final iteration of each experiment. Finally, we analyze the performance attained by several classifiers in validation experiments and summarize the experimental findings.

In the first iteration the datasets used to train the CS system are equal for all experiments. As previously mentioned, a total of 45000 images is used for training while 5000 are used for testing purposes. Table 5.4 summarizes the results attained. The performance in training is close to perfection for the three experiments, FULL misclassifies 2 out of 45000 training instances, FW 1 and BW 0 (a perfect score). In the test instances all classifiers performed flawlessly, indicating that they generalized correctly. In this iteration the BW classifier uses 45 features while the FW classifier uses 30. Indicating that a small number of features is sufficient to correctly identify all training and test instances.

The obtained classifiers are used to guide the evolutionary runs of each experiment. In all experiments, in during the first iteration of the boosting algorithm, the EC system was able to generate individuals that were classified as external. Figures 5.6, 5.7 and 5.8 present the fittest individual of each generation of the GP engine during the first iteration for each of the experiments. The first number indicates the generation number of the individual and “val” indicates its fitness.

Table 5.4: Performance in the first iteration.

		Training		Test	
model	#features	#Correct	#Incorrect	#Correct	#Incorrect
FULL	804	44998	2	5000	0
FW	30	44999	1	5000	0
BW	45	45000	0	5000	0

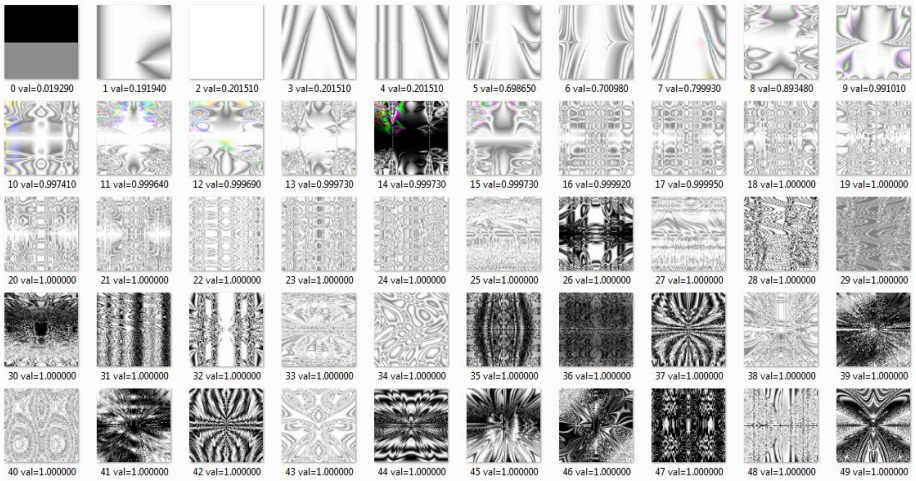


Figure 5.6: Fittest individual of each generation of the FULL experiment in the first iteration.

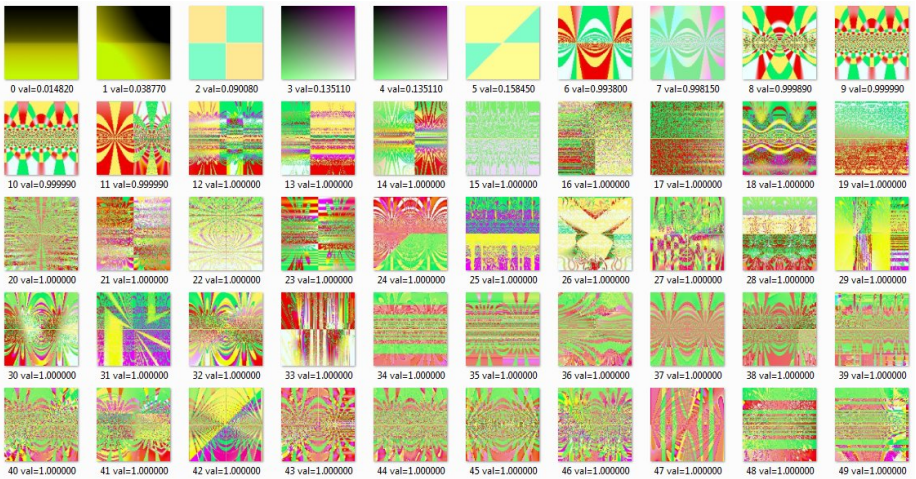


Figure 5.7: Fittest individual of each generation of the FW experiment in the first iteration.

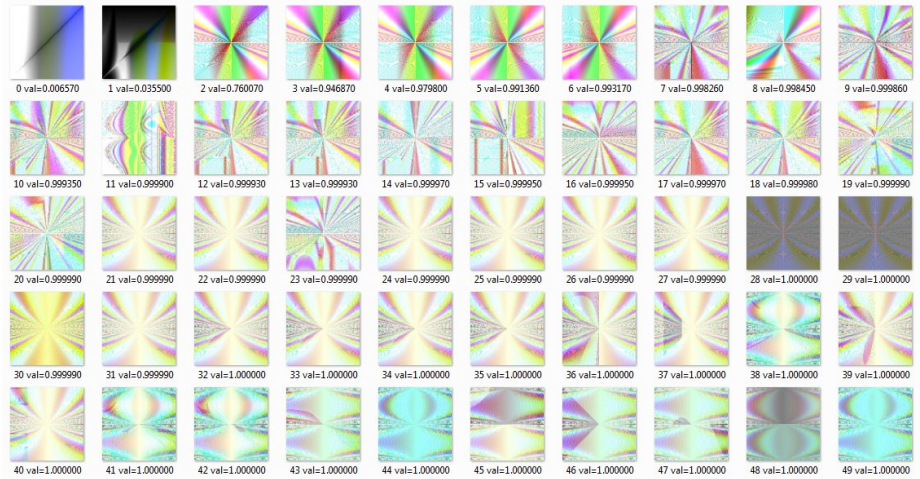


Figure 5.8: Fittest individual of each generation of the BW experiment in the first iteration.

As would be expected, the different experiments converged to different types of imagery. The FULL experiment converged to greyscale images while the FW experiment converged colored ones. In spite of this superficial difference, there is a common trend between these experiments: they both converged to highly complex, "noisy" images. This is an expected result. With the used function set it is unlikely that NEvAr produces, randomly, complex, noisy images. Consequently initial internal set is composed, mainly, of simplistic images, which makes the evolution of complex ones an obvious way to misguide the CS system. The BW experiment appears to have converged to images that are characterized simultaneously by (i) a complex and noisy "background" and by (ii) low contrast and high luminosity. The converge to this type of imagery can also be explained by the unlikelihood of randomly creating NEvAr images with these properties, and hence their absence from the initial internal set.

In a nutshell, like in Machado et al. [59] more than evolving images that resemble painting the system is, apparently, evolving images that are atypical of NEvAr and hence cause classification errors.

It is also visible from the presented figures that, in the first iteration, all experiments were able to find images classified as external in relatively few generations, 6 or less, and that all experiments converged to images considered false positives.

We could present charts showing the evolution of fitness during the course of the runs, however these charts would be (i) pointless, because our focus is the analysis of the performance of the classifier, not the analysis of the evolutionary process; (ii) meaningless, because they would lack statistical validity since only one run is performed for each iteration. We could make 30 different repetitions of each experiment, but we lack the resources to do it and – unless we wished to focus on the evolutionary aspect – there is very little to gain from this.

The individuals evolved during the first iteration are added to the internal set images, replacing the randomly generated ones, and a new iteration is started. Since we only add images that are unique and that the number of unique images

Table 5.5: FULL's performance during training.

Iteration	%Correct	TP	FN	FP	TN	Recall	Precision
0	99.99556	22498	2	0	22500	0.99991	1.00000
1	99.99333	22498	2	1	22499	0.99991	0.99996
2	99.99778	22499	1	0	22500	0.99996	1.00000
3	99.99556	22498	2	0	22500	0.99991	1.00000
4	99.99778	22499	1	0	22500	0.99996	1.00000
5	99.99778	22499	1	0	22500	0.99996	1.00000
6	99.98222	22493	7	1	22499	0.99969	0.99996
7	99.99333	22497	3	0	22500	0.99987	1.00000
8	99.99556	22498	2	0	22500	0.99991	1.00000
9	99.99333	22498	2	1	22499	0.99991	0.99996
10	99.99333	22498	2	1	22499	0.99991	0.99996
11	99.98444	22493	7	0	22500	0.99969	1.00000
12	99.98889	22495	5	0	22500	0.99978	1.00000
13	99.99333	22497	3	0	22500	0.99987	1.00000

varies from experiment to experiment, the number of iterations varies from experiment to experiment. FULL performed 12 iterations, FW 10 and BW 11 before reaching the termination criterion of replacing 22500 individuals of the internal set.

It would tedious and space consuming to present snapshots of all iterations performed. A such, since our primary interest is classifier performance, we chose to focus on the performance of the classifiers, the variations in the number of selected features and the number of misclassified images generated in each evolutionary run.

First we present the performance in training and test in terms of percentage of correctly classified examples and its confusion matrix values: TP – true positives, TN - true negatives , FP – false positives and FN – false negatives. In this context we consider *external* images as belonging to the *positive* class and *internal* images as belonging to the *negative* class. Thus, a FP is a internal image that is classified as external.

Additionally, to evaluate the performance of the experiments throughot the iterations two statistical measures are used: precision and recall. The precision consists in the capability of a classifier to recognize real positive examples and its measure by $TP/TP + FP$. Recall measures the ability of the classifier in detecting all positives by computing $TP/TP + FN$. These measures were selected because both reflect how the classifier responds towards the positive examples that these experiments pretended to exploit by creating false positives.

Tables 5.5, 5.6 and 5.7 show the performance of the classifiers across iterations during in the training set.

These training results show that, in training, the FULL approach has the best performance in terms of FP and FN. BW has the second best performance, while FW has the worst performance. The percision and recall measures follow the same overall trend. The performance of the classifier in the FULL approach is consistent across iterations, while the performance of FW and BW suffers several fluctuations. This result can be explained by the strategies adopted in these two experiments so select features, which does not ensure optimality. The performance in the test sets is displayed in tables 5.8, 5.9 and 5.10. Overall, the results attained in training are similar to those attained in the test sets, both in terms of overall trend and in terms of percentage of errors. Across all

Table 5.6: FW's performance during training.

Iteration	%Correct	TP	FN	FP	TN	Recall	Precision
0	99.998	22499	1	0	22500	0.999956	1.0000
1	99.891	22469	31	18	22482	0.998623	0.9992
2	99.704	22442	58	75	22425	0.99742	0.9967
3	99.949	22497	3	20	22480	0.999867	0.9991
4	99.000	22246	254	196	22304	0.98874	0.9913
5	98.687	22088	412	179	22321	0.981877	0.9920
6	99.567	22369	131	64	22436	0.994195	0.9972
7	98.898	22151	349	147	22353	0.984627	0.9935
8	99.562	22317	183	14	22486	0.991927	0.9994
9	99.851	22452	48	19	22481	0.997869	0.9992
10	99.958	22481	19	0	22500	0.999156	1.0000
11	99.973	22494	6	6	22494	0.999733	0.9997

Table 5.7: BW's performance during training.

Iteration	%Correct	TP	FN	FP	TN	Recall	Precision
0	100.00000	22500	0	0	22500	1.00000	1.00000
1	100.00000	22500	0	0	22500	1.00000	1.00000
2	99.97556	22489	11	0	22500	0.99951	1.00000
3	99.90222	22461	39	5	22495	0.99827	0.99978
4	99.62222	22447	53	117	22383	0.99764	0.99480
5	99.96667	22497	3	12	22488	0.99987	0.99947
6	99.95556	22486	14	6	22494	0.99938	0.99973
7	99.94889	22487	13	10	22490	0.99942	0.99956
8	99.96222	22494	6	11	22489	0.99973	0.99951
9	99.98000	22494	6	3	22497	0.99973	0.99987
10	99.99333	22497	3	0	22500	0.99987	1.00000
11	99.97111	22492	8	5	22495	0.99964	0.99978
12	99.97556	22492	8	3	22497	0.99964	0.99987

Table 5.8: FULL's performance during test.

Iteration	%Correct	TP	FN	FP	TN	Recall	Precision
0	100.00000	2500	0	0	2500	1.00000	1.00000
1	100.00000	2500	0	0	2500	1.00000	1.00000
2	100.00000	2500	0	0	2500	1.00000	1.00000
3	100.00000	2500	0	0	2500	1.00000	1.00000
4	100.00000	2500	0	0	2500	1.00000	1.00000
5	100.00000	2500	0	0	2500	1.00000	1.00000
6	100.00000	2500	0	0	2500	1.00000	1.00000
7	100.00000	2500	0	0	2500	1.00000	1.00000
8	100.00000	2500	0	0	2500	1.00000	1.00000
9	100.00000	2500	0	0	2500	1.00000	1.00000
10	100.00000	2500	0	0	2500	1.00000	1.00000
11	100.00000	2500	0	0	2500	1.00000	1.00000
12	100.00000	2500	0	0	2500	1.00000	1.00000

Table 5.9: FW's performance during test.

Iteration	%Correct	TP	FN	FP	TN	Recall	Precision
0	100.00000	2500	0	0	2500	1.00000	1.00000
1	99.90000	2498	2	3	2497	0.99920	0.99880
2	99.70000	2496	4	11	2489	0.99840	0.99560
3	99.94000	2500	0	3	2497	1.00000	0.99880
4	99.04000	2476	24	24	2476	0.99040	0.99040
5	98.38000	2448	52	29	2471	0.97939	0.98840
6	99.62000	2487	13	6	2494	0.99481	0.99760
7	99.02000	2465	35	14	2486	0.98612	0.99440
8	99.58000	2481	19	2	2498	0.99245	0.99920
9	99.82000	2493	7	2	2498	0.99721	0.99920
10	100.00000	2500	0	0	2500	1.00000	1.00000
11	100.00000	2500	0	0	2500	1.00000	1.00000

iteration and and experiments precision and recall values above 0.99 were always attained.

Table 5.11 displays the number of FP generated by the evolutionary engine across iterations for each experiment. As it can be observed in the FW and BW experiments the evolutionary engine was able to find FP in all iterations. In the FULL experiment the EC engine was unable to find FP from the 5th iteration onwards and was also unable to find FP in the 2nd iteration. This inability to find FP has consequences in terms of analysis and requires explanation.

One of the obvious consequences is the following, in the FW and BW experiments the task of the classifier becomes, arguably, harder from iteration from iteration. The same does not happen for the FULL experiment since no FP, i.e. hard to classify images, are added to the internal set. This can contribute, to the the better performance of the FULL approach in test and training across iteration. Nevertheless, as we will se later, it does not appear to be the only reason why the performance is better.

It is also important to understand why the evolutionary engine does not find FP. As previously mentioned NEvAr can generate any image, so it is theoretically possible to generate FP. Additionally, as we will show in the next section, not only it is theoretically possible, images that are classified as FP by these classifiers have been evolved with NEvAr in the scope of other experiments.

Thus, all evidence points to the following explanation, the fitness landscape resulting from the use of this classifiers makes evolution difficult. An analysis of

Table 5.10: BW's performance during test.

Iteration	%Correct	TP	FN	FP	TN	Recall	Precision
0	100.00000	2500	0	0	2500	1.00000	1.00000
1	100.00000	2500	0	0	2500	1.00000	1.00000
2	100.00000	2500	0	0	2500	1.00000	1.00000
3	99.94000	2499	1	2	2498	0.99960	0.99920
4	99.60000	2495	5	15	2485	0.99799	0.99400
5	99.96000	2500	0	2	2498	1.00000	0.99920
6	99.96000	2499	1	1	2499	0.99960	0.99960
7	99.90000	2499	1	4	2496	0.99960	0.99840
8	99.96000	2500	0	2	2498	1.00000	0.99920
9	99.98000	2500	0	1	2499	1.00000	0.99960
10	99.98000	2499	1	0	2500	0.99960	1.00000
11	99.94000	2499	1	2	2498	0.99960	0.99920
12	99.98000	2499	1	0	2500	0.99960	1.00000

Table 5.11: False positives generated in the experiments per iteration. Note that the Fw experimnt stops after 10 iterations while the BW experiment stops after 11.

Iterations	FULL	FW	BW
0	1616	1165	1451
1	0	1081	1091
2	598	1575	1074
3	393	840	1261
4	73	832	409
5	0	1755	441
6	0	1110	613
7	0	651	499
8	0	271	292
9	0	70	484
10	0	1184	1458
11	0	-	995
12	0	-	-
Total	2680	10534	10068

the values returned by the fitness function reinforces this explanation: there are no intermediate values, only values above 0.7 or bellow 0.3 were returned, even when the classifier is tested with external imagery. In most cases, and although the tolerance threshold is set to 0.3, the values are bellow 0.1 or above 0.9. As a consequence, the evolutionary algorithm would only be able to find a FP by chance, and this appears to be the main reason for the lack of FP. The BW and FW classifiers behave differently, returning intermediate values. This appears to indicate that in the considered experimental conditions a high number of features tends to result in a discontinuous output, which hinders evolution. The confirmation of this interpretation requires further experimentation that is beyond the scope of this dissertation.

Table 5.12 shows the number of features used by the classifiers in each iteration for the FW and BW experiments. Although variations occur the number of features used is significantly lower than the total number of features available (804) indicating that good performance is attainable with a relatively low number of features. As would be expected, FW tends to use a smaller number of features than BW selection. Correlating these results with the performance across experiments in training and test sets, the experiments indicate that there

Table 5.12: Number of features selected per iteration.

Iterations	FW	BW
0	30	45
1	22	138
2	19	98
3	37	97
4	11	105
5	11	48
6	11	55
7	7	53
8	15	53
9	8	58
10	25	43
11	35	34
12	-	34
Average	19.25	66.23077

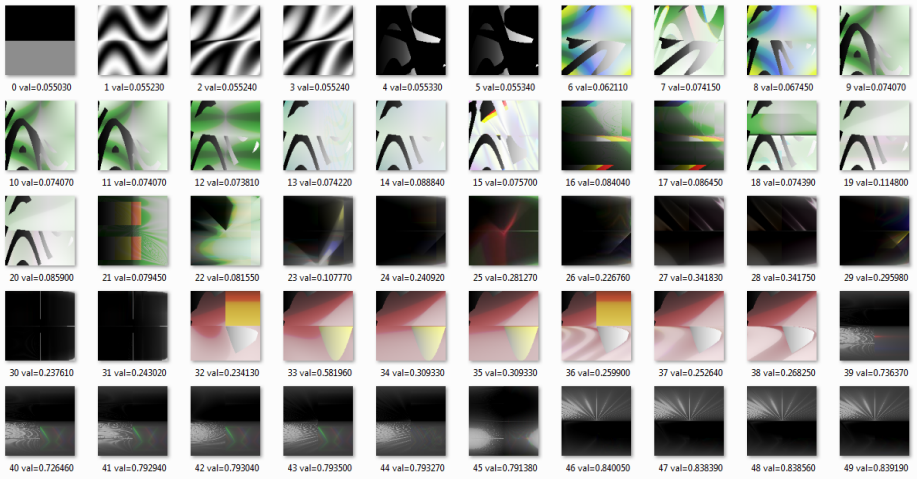


Figure 5.9: Fittest individual of each generation of the FULL experiment in the 4th iteration, the last were FP were found.

is a positive correlation between the number of features used by the classifier and the performance of the classifier.

Figures 5.10, 5.10 and 5.11 present the fittest individual of each generation of the GP engine during the last iteration where FP were found for each of the experiments. As would be expected these images differ, stylistically, from the ones evolved in the first iteration (and also from the ones evolved in intermediate ones). It is interesting to notice that, even at this stage, and in spite of the high classifier performance, the EC engine is still able to find relatively simple images that are misclassified.

Once the FULL, FW and BW experiments were finished we trained classifiers using the images that were added to the initial set in the last run (note that these classifiers were trained but not used to guide an evolutionary run, since the experiments were finished). Table, the table 5.13 summarizes the performance attained by these classifiers. As it can be observed the number of incorrectly classified instances is residual for all experiments.

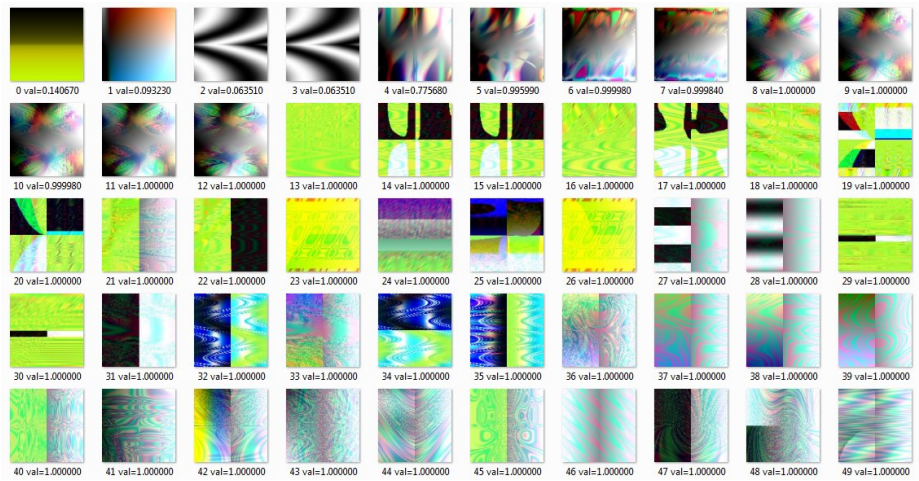


Figure 5.10: Fittest individual of each generation of the FW experiment in the last iteration.

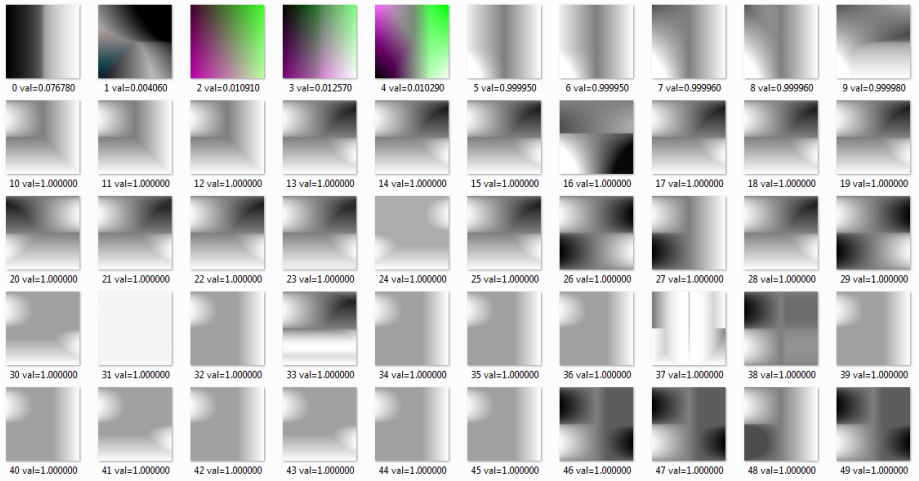


Figure 5.11: Fittest individual of each generation of the BW experiment in the last iteration.

Table 5.13: Performance in the last iteration.

		Training		Test	
model	#features	#Correct	#Incorrect	#Correct	#Incorrect
FULL	804	44997	3	5000	0
FW	35	44988	12	5000	0
BW	34	44989	11	4999	1

Table 5.14: Percentage of images correctly classified during training and test in a models versus Data sets test.

	initial		last BW		last FW		last FULL	
Model	Train	Test	Train	Test	Train	Test	Train	Test
BWi	100	100	99.99	100	99.99	100	99.99	100
BWl	99.98	100	99.97	99.99	99.92	99.94	98.87	98.80
FWi	99.99	100	99.99	100	99.99	100	99.99	100
FWl	100	100	99.99	100	99.97	100	99.43	99.44
FULL	99.99	100	99.99	100	99.99	100	99.99	100

5.5 Assessing Classifier Performance

In order to analyze the influence of the selected features and of the selected training sets we create several different classifiers varying these parameters. We consider 5 sets of features:

BWi – The set of features selected by the BW experiment in the initial iteration;

BWl – The set of features selected by the BW experiment in the final iteration;

FWi – The set of features selected by the FW experiment in the initial iteration;

FWl – The set of features selected by the FW experiment in the final iteration;

FULL – The full set of features 804 features;

Four training and test datasets are considered:

Initial – The initial dataset of internal and external images which is used in all experiments;

Last BW – The final dataset of internal and external images resulting from the BW experiment;

Last FW – The final dataset of internal and external images resulting from the FW experiment;

Last FULL – The final dataset of internal and external images resulting from the FULL experiment;

Considering all combinations results in 20 distinct models. Table 5.14 present the results attained in testing and training by each of them.

The experimental results presented so far concern the performance of the classifiers with training sets closely related with their training. As such they do not offer much information regarding which classifiers perform better nor regarding how the boosting process affected classifier performance. The goal of the approach is to improve classifier performance by synthesizing images that are misclassified and including these images in the training set.

To assess classifier performance we compiled four independent validation sets, which are not directly related with the images used in training, as follows:

1. Paintings – 2408 images retrieved from Flickr using the keyword “painting”;

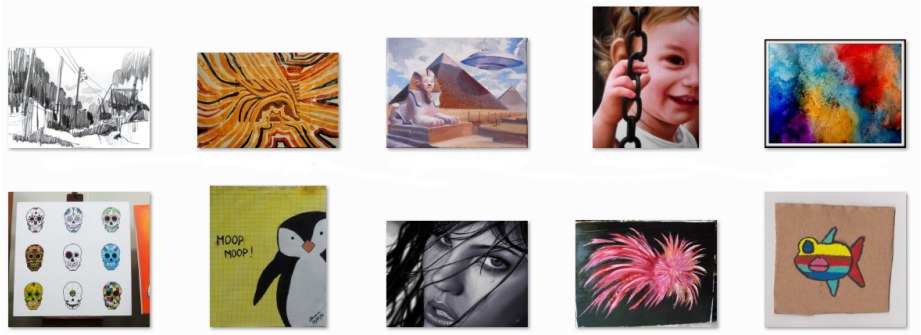


Figure 5.12: Samples from Paintings validation set.

2. Jornadas – 2096 images generated using user-guided evolution on NEvAr by a group of users;

3. Compilation - 8320 images generated from user-guided evolution runs of NEvAr used by Machado et al. in previous works [57, 58, 59];

4. Fractals – 2836 images retrieved from Flickr using the keyword "fractal";

The first validation dataset is intend to assess the performance of the models in the evaluation of external imagery, in this case paintings. It is important to notice that the retrieved files are not related with the ones using in the training and testing of the CS models. They come from a different source and the considered paintings are different from the ones using in training and testing. Figure 5.12 presents some instances of this dataset.

The second and third validation datasets are intended to assess the performance of the models in the classification of internal image. Both are composed of images created with NEvAr in interactive evolution runs that were valued by the users (i.e. that had fitness scores above the default score of zero). The difference between them is that the images that belong to "Jornadas" were created in a single afternoon by a group of users that were not familiar with the tool, while the images of "Compilation" were evolved by Penousal Machado over several years of experimentation with NEvAr. Figures 5.13 and 5.14 present samples of these validation datasets.

The last dataset is composed of computer generated images, mainly fractal art. These image fall between the considered categories. They are external in the sense that they were not created with NEvAr, although they could have been, however they are not paintings. Does, deciding which is the correct classification in this case is debatable. The inclusion of this validation dataset was motivated by our curiosity and also because the variations in performance could possibly give us interesting insights.

Table 5.15 displays the results attained by the different models. For the purpose of this and the following tables we consider that the correct classification of fractal images is "external", but we present average results with and without considering the fractal validation set to avoid biasing the results.

Table 5.16 summarizes these results presenting the average percentage of correctly classified images for the five different feature sets used by the models. Underlined numbers indicate the lowes performance while bold entries indicate

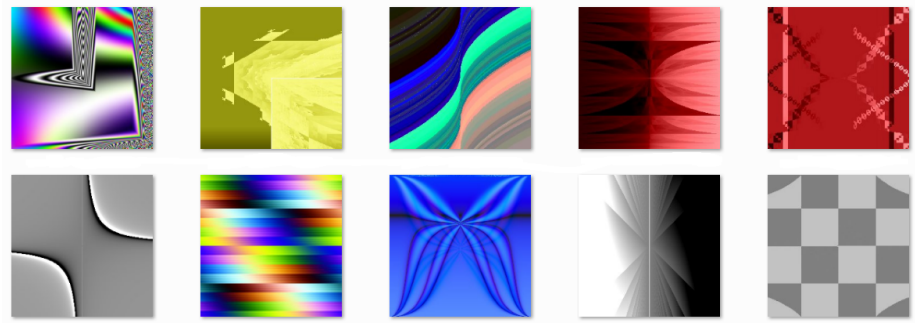


Figure 5.13: Samples from Jornadas validation set.

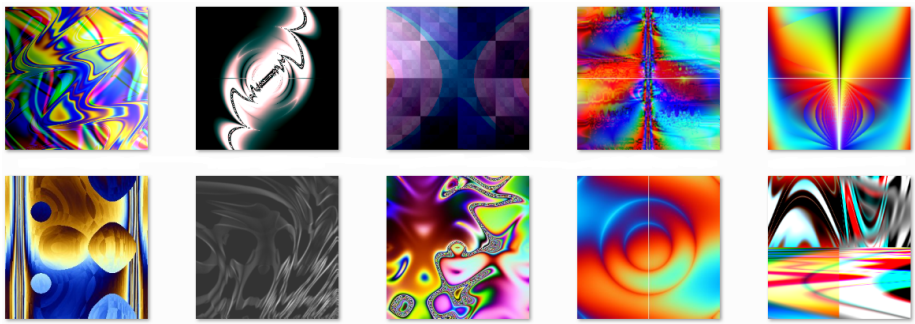


Figure 5.14: Samples from Compilation validation set.

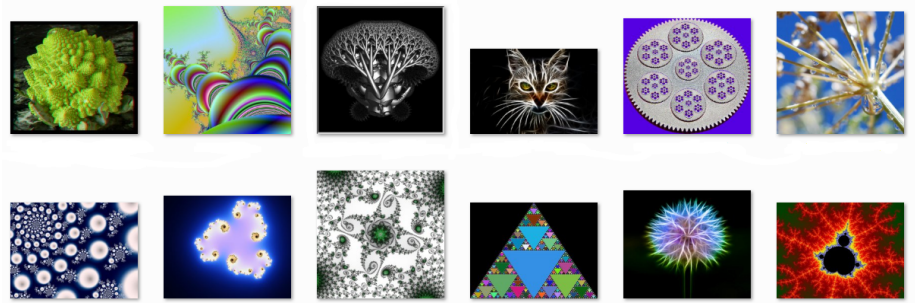


Figure 5.15: Samples from Fractal validation set.

Table 5.15: Percentage of correctly classified images of the validation sets.

Features	Dataset	Paintings	Compilation	Jornadas	Avg.	Fractals	Avg.
BWi	Initial	98.28	51.55	74.58	74.80	93.07	79.37
	Last BW	97.73	68.47	80.41	82.21	86.74	83.34
	Last FW	98.11	59.35	79.65	79.03	90.95	82.01
	Last FULL	99.50	38.30	66.32	68.04	96.71	75.20
BWl	Initial	98.49	52.78	77.21	76.16	92.65	80.28
	Last BW	97.14	69.59	86.62	84.45	83.70	84.27
	Last FW	95.97	71.76	88.58	85.43	87.06	85.84
	Last FULL	92.82	78.91	93.07	88.27	75.47	85.07
FWi	Initial	98.82	44.79	72.96	72.19	94.13	77.68
	Last BW	98.53	49.46	72.43	73.48	94.24	78.67
	Last FW	98.49	53.58	74.53	75.53	93.00	79.90
	Last FULL	99.29	34.81	64.74	66.28	97.70	74.13
FWl	Initial	98.78	51.06	77.88	75.91	93.25	80.24
	Last BW	97.86	68.38	84.95	83.73	82.26	83.36
	Last FW	96.60	78.73	90.11	88.48	85.47	87.73
	Last FULL	94.37	81.00	91.97	89.12	77.24	86.15
FULL	Initial	99.08	60.17	78.69	79.31	93.78	82.93
	Last BW	99.20	62.08	80.60	80.63	94.06	83.99
	Last FW	99.03	66.03	81.99	82.35	93.50	85.14
	Last FULL	99.62	24.98	58.48	61.03	97.95	70.26
Average		97.88	58.29	78.79	78.32	90.15	81.28

the highest performance. The average success rates in the classification of the Paintings dataset vary between 96.10 and 99.23. The success rates in the classification of the Compilation and Jornadas datasets are significantly lower and more variable. Two striking observations should be made:

- Correctly classifying Compilation images appears to be more difficult than correctly classifying Jornadas images. This result was expected and is easily explainable, several of the images included in the compilation dataset result from long and time consuming runs, as such they deviate from typical NEvAr imagery a lot more than the Jornadas images that result from relatively short runs.

- BWl and FWl consistently perform better than BWi and FWi in these validation datasets. This indicates that the features selected in the final iterations are more suitable to correctly identify internal generated imagery without large losses of performance in the classification of external imagery. Additionally, BWl and FWl also perform better than FULL in these datasets.

Taking into account the results attained in the fractal dataset does no significantly change these conclusions. Nevertheless, it is interesting to notice that the initial sets of features, BWi and FWi, tend to classify more Fractal images as external than BWl and FWl. Like the previous results, this was expectable. In some sense, BWl and FWl, are more prune to classify images as internal than BWi, FWi or FULL.

Table 5.16 summarizes the results by presenting the average percentage of correctly classified images per dataset used in training the models. The most striking observation is that the last FULL training dataset yields the worst average performance for all scenarios, except fractal imagery.

Using the FULL feature set, the FWi feature set and the BWi feature set in conjunction with the Last FULL training set yields bad performance in all sets, most notably in Compilation and Jornadas. Surprisingly using the FWl feature set or the BWl and Last Full for training yields the best performances in Jornadas and Compilation.

The results attained with the Last FULL training set require explaining. Although conclusive it is impossible to draw decisive conclusions without further testing we advance a tentative explanation.

From the 5th iteration onwards the FULL experiment was unable to find FP. Therefore, from that iteration on, the initial internal images, that had been randomly generated with NEvAr, were replaced by internal imagery generated during the course of the run. Although the runs were unsuccessful in finding false positive, they did converge to suboptimal solutions. Consequently, by replacing the randomly generated imagery with the images created during the experiment one lost diversity and the internal dataset became less representative of what NEvAr could achieve than the initial internal dataset. Oddly this explains both the bad and performances attained in the internal datasets, Compilation and Jornadas.

When the Last Full training set is used in conjunction with the FULL feature set it results in the highest overall performance in the paintings dataset, and also in the lowest overall performance in Jornadas and Compilation, the set is less of what NEvAr can achieve and hence when confronted with new images the classifier "assumes" they were not generated with NEvAr. In other words the classifier is biased towards identifying the images as external, this is can be partially confirmed by the high performance attained in the Paintings and Fractal validation sets for this combination. The results attained BY using BWi and FWi in conjunction with Last Full can be explained in a similar way. When combining FWl and BWl with Last Full we have the inverse scenario, best performance in Compilation and Jornadas and Worst performance in Paintings and Fractals. Thus, it appears that the classifier is now balanced towards the classification of images as Internal. Our explanation for this tendency is that having a set that is not representative of what can be attained is, in some way, similar to having a set that is not representative of what cannot be /easily) attainable, and as such may result in biased classifiers. The reason why the Full and BWi and FWi feature sets yield behaviors that are opposite to those displayed by BWl and FWl appears to be the following, as previously mentioned BWl and FWl promote correct classification of internal imagery, in the presence of representative training sets they are able to increase the performance in the classification of internal imagery without significantly hindering the performance in the recognition of external imagery. When the internal image training set becomes less representative a tendency to classify images as internal is developed.

As it can be observed, all training datasets result in high success rates in the recognition of paintings. However, the same is not true for the remaining validation sets. In the Jornadas and Compilation validation set one can observe increases of performance when comparing the initial training set with last BW and last FW experiments. By using Last BW for training one attains an average performance in the paintings dataset of 98.09% which is only .6% lower than the one attained when using the Initial set, however the performance in Comilation is 11.53% higher and in Jornadas 4.74% higher. Using Last FW for training yields

Table 5.16: Average percentage of correctly classified images per validation set. Results grouped by the features used by the models.

Experiment	Paintings	Compilation	Jornadas	Average	Fractals	Average
BWi	98.40	54.42	75.24	76.02	91.87	79.98
BWl	96.10	68.26	**86.37**	83.58	84.72	83.86
FWi	98.78	45.66	71.17	71.87	94.77	77.59
FWl	96.90	**69.79**	86.23	**84.31**	84.55	**84.37**
FULL	**99.23**	53.32	74.94	75.83	**94.82**	80.58

Table 5.17: Average percentage of correctly classified images per validation set. Results grouped by dataset used to train the model.

Data set	Paintings	Compilation	Jornadas	Average	Fractals	Average
Initial	**98.69**	52.07	76.26	75.67	**93.38**	80.10
Last BW	98.09	63.60	81.00	80.90	88.20	82.72
Last FW	97.64	**65.89**	**82.97**	**82.17**	90.00	**84.12**
Last FULL	97.12	51.60	74.92	74.54	89.01	78.16

an average performance that is 0.95% lower in the Paintings validation set, but 13.82% higher in Compilation and 6.71% higher in Jornadas. These results indicate that our framework was able to produce training sets that increase the performance in the recognition of internal imagery without significantly hindering the performance in the recognition of external imagery.

In figures 5.16 to 5.19 we present some of the images that are most often misclassified. It is particularly interesting to observe that some of the most frequently misclassified paintings are either not paintings at al, or are, to say the least, atypical ones. In what concerns the frequently misclassified Fractal images, it is easy to understand why they could be confused with NEvAr images. In fact, as previously discussed considering this images as external or internal is highly subjective. In what regards the frequently misclassified Jornadas and Compilation images, the common characteristic is that they are atypical in relation with the imagery usually produced by NEvAr.

5.6 Summary

The conducted experiments indicate that it is possible to improve classifier performance using the boosting approach proposed in this thesis. Overall the experimental results suggest that in the considered experimental settings this is accomplishable in two different ways:

- By changing the training datasets by synthesizing new instances, in par-

Figure 5.16: Examples of misclassified images from Paintings set.

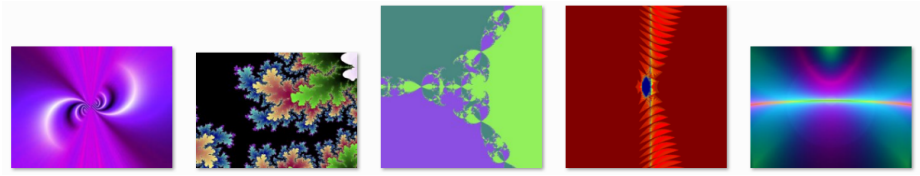


Figure 5.17: Examples of misclassified images from Fractals set.

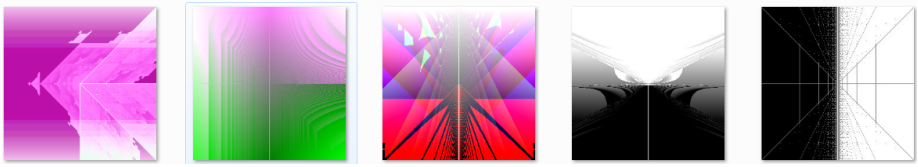


Figure 5.18: Examples of misclassified images from Jornadas set.

ticular false positives, created during the course of the runs;

- By using the boosting framework to select new combinations of features;

Although the results are not entirely conclusive, they can be considering promising and demonstrate the potential of the proposed framework. Further experimentation is required and is already taking place. We are particularly interested in: (i) Confirming the experimental findings via further testing and analysis (ii) testing alternative image replacement schemes (e.g. only adding images that were misclassified) (iii) Producing experiments with an higher number of iterations (iv) Testing alternative feature selection schemes.

Figure 5.19: Examples of misclassified images from Compilation set.

Chapter 6

Improving Face Detection

The current Chapter describes the second test to the framework presented in Chapter 4. This framework was explored in an attempt to lessen the face detection shortcomings identified in the preliminary experiment described in (section 3.1).

We begin by making an introduction, stating the problems encountered and the experiment's context. Next, in section 6.2, the approach is defined and the experimental setup described in section 6.3. Then the experimental results are presented and analyzed in section 6.4. Finally we perform validation tests, in section 6.5, and we summarize the results.

6.1 Introduction

Object detection systems, in particular face detection, have become a topic of interest and research. Nowadays applications that employ these kind of systems are also becoming very popular. For instance, they can be found in search engines, social networks, incorporated in cameras, or in applications for smart phones. Most, if not all, of these systems have shortcomings which may result in false alarms, the detection of faces in images were they are absent. For illustration purposes, some examples are presented in figures 6.1, 6.2 and 6.3. Figure 6.1 presents false positives "faces" filter in Google image search. While some of the false positives are acceptable, other are hard to understand. While cataloguing some images with Picasa, figure 6.2, similar results were attained. Tests with the iPhone and the "Self photo" application revealed the same behavior (see figure 6.3). In summary, although most of the images are catalogued correctly is trivial to find several false positives, it was trivial to find false positives for all face detection applications we tested.

In the experiment described in section 3.1 an attempt to evolve images that could resemble human was made. Some of the results were arguably reasonable but most of the generated images were false positives that hardly resemble a human face. The research described in this chapter intends to explore the framework proposed in chapter 4 in order to improve the performance of a face detection algorithm.

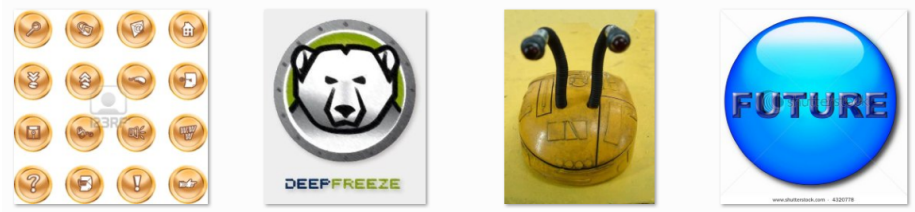


Figure 6.1: Sample of False positives retrieved via Google Image search using the "computer" and the google "faces" filter.

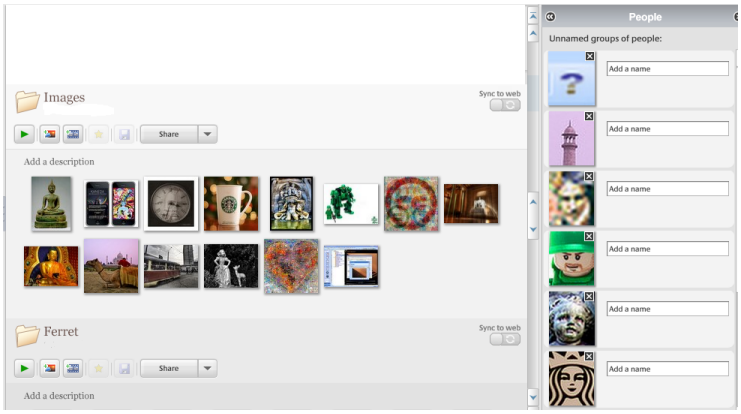
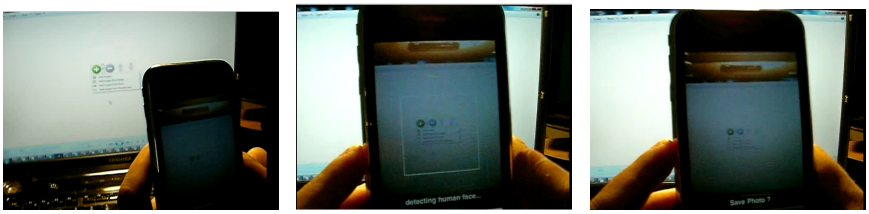


Figure 6.2: False alarms while using Picasa (Unnamed groups of people).



(a) (b) (c)

Figure 6.3: False positives detected while using "Self Photo" application for IPhone.

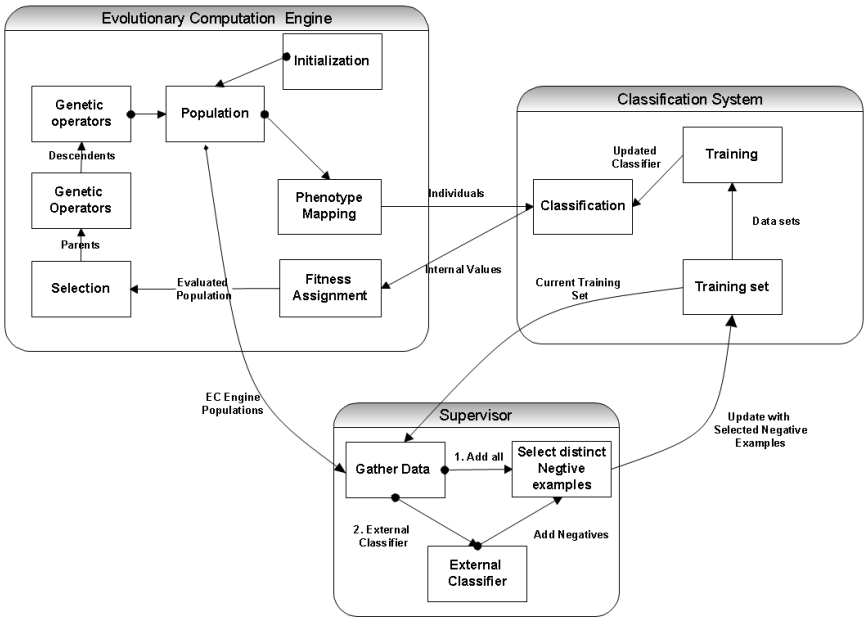


Figure 6.4: System overview.

6.2 The Approach

The approach is similar to the one described in chapter 5 with a few modifications. It can be define by these steps:

1. Selection of a positive and negative image set; In this context positives are images that contain faces while negatives are images were no face is present;
2. The Classifier System (CS) is built to distinguish these initial sets;
3. A new EC run is started. Output values from the CS are used to assign fitness;
4. The EC run stops when a termination criterion is met (e.g., a pre-established number of generations, attaining a fitness value);
5. The set of negative images is updated by the Supervisor which uses the generated images from the EC run to chose individuals to add;
6. The process is repeated from step 3 until the boosting criterion is met;

As it can be observed in figure 6.4 the changes to the conceptual model are few, and only concern the Supervisor module,

As previously we use NEvAr's genetic engine, already mentioned in Chapters 3 and 5, for evolution purposes. As previously the goal is of the evolutionary runs is to evolve images that are classified as containing faces and fitness is assigned

Table 6.2: Haar Training parameters.

Parameter	Setting
features	ALL
Input width	20
Input height	20
Number of stages	20
Number of splits	2
Min Hitrate	0.99
Max False Alarm	0.5
Adaboost Algorithm	GentleAdaboost

according to the outputs of the CS system valuing images that resemble faces. The formula used is similar to the one presented in section 3.1.

The CS system consists in a Haar Cascade classifier [86].

The Supervisor for this experiment can operate in two modes: *addneg* or *classif*. The *addneg* mode consists in adding all images created during the EC runs that were classified as faces to the negative set. The *classif* mode uses an external classifier to filter the images that are added to the negative set. The image is only added if the external classifier does not identify a face in the image. Thus if the classifiers disagree. The chosen external classifier is from fdlib [1] that is based on a cascade of Reduced Support Vectors[40]. Its fast computation and easy integration with the experiment were the determinant aspects for this choice. As in previous experiments, the fitness depends on the outputs of the CS.

6.3 Experimental Setup

NEvAr's settings are presented in table 6.1 and are similar to those used in Chapter 5, the difference is that here we use a population size of 100 resulting in 5000 individuals at the end of the EC run. The random number seed in all stochastic methods is always restarted, for example, at the start of the EC run the start population is always the same for all boosting iterations. The starting population are presented in figures B.2 and B.3.

Table 6.1: NEvAr parameters.

Parameter	Setting
Population Size	100
Number of generations	50
Crossover probability	0.8
Mutation operators	sub-tree swap, sub-tree replacement, node insertion, node deletion, node mutation
Initialization method	ramped half-and-half
Initial maximum depth	5
Mutation max tree depth	3
Function set	+, -, *, /, min, max, abs, sin, cos, if, pow, mdist, warp, sqrt, sign, neg
Terminal set	X, Y, scalar and vector random constants

An overview of the Haar cascade classifier approach was made in section 3.1. For training purposes we used the OpenCV "opencv_haartraining" tool. The

[1] fdlib – http://people.kyb.tuebingen.mpg.de/kienzle/fdlib/fdlib.htm

Table 6.3: Internal classifier parameters.

Parameter	Setting
Window width	20
Window height	20
Scale factor	1.1
Min face width	$0.75 \times inputwidth$
Min face height	$0.75 \times inputheight$

relevant classifier parameters can be consulted in table 6.2 and are were chosen based on the work of Lienhart [50] and Viola [86]. According to Lienhart work, the minimum hit rate and maximum false alarm rate expected, while using the parameters described on table 6.2 are, approximately, of $0.99^{20} \simeq 0.818$ and $0.50^{20} \simeq 9 \times 10^{-7}$, respectively. Although while using this training, the classifier may reach those values, the training time may take days, weeks or months. Another set of parameters addresses this issue, providing faster training and similar performance. For instance, one can define the maximum number of stages to be trained. Other influential parameters are the input pattern size ($Inputwidth \times Inputheight$) and the number splits, that define the stage classifier maximum number of feature splits that allow combinations of features inside of the stage creating a tree of features. In the experiments conducted in this Chapter the input pattern chosen had a size of 20×20 pixels and the number of splits was 2. The selected Ada-boost was GentleAdaboost. This combination of parameters had good hit and false alarm rates according to the study conducted by Lienhart et al.[49].

In addition to the training parameters, there are other classifier settings that need to be established. All parameters used are based on OpenCV default parameters, except the minimum face width and height, scale factor and window size, which are presented in table 6.3. The same parameter settings were employed in Chapter 3.1.

The external FDlib classifier, used for supervision, only has a tolerance threshold and the default value was kept.

In order to test the different classifiers a performance evaluation tool was implemented. It allows loading an image test set, with a ground truth file associated, and a classifier configuration file. The performance is measured in terms of hit (H), miss (M), false alarm(FA), correct and incorrect. In order to do this, it loads the parameters and classifier of the configuration file, and perform the face detection. Then it compares with the ground truth file, if the result match or lays in the tolerance area defined by the performance tool parameters, it is a hit. If it lays outside the tolerance area it is counted as a false alarm. In case of not detecting a face, if it exists, it is counted has a miss. The parameters are defined in table 6.4 and are based on the default parameters of OpenCV's "opencv_performance" tool. This tool was implemented because the performance tool of OpenCV only calculates the performance of positive examples and accepts only cascade classifiers.

The quality of the positive and negative datasets used in training influences significantly the performance of a classifier. It is important to have good positive examples of object that we are training in order to attain good success rates. For this experiments part of two referenced image datasets were used: "The Yale Face Database B" ([23]) and "BioID Face Database"([36]). "The Yale Face

Table 6.4: Parameters used by the performance tool.

Parameter	Setting
Window width	20
Window height	20
Scale factor	1.1
Maximum size difference factor	1.5
Maximum position difference factor	0.5

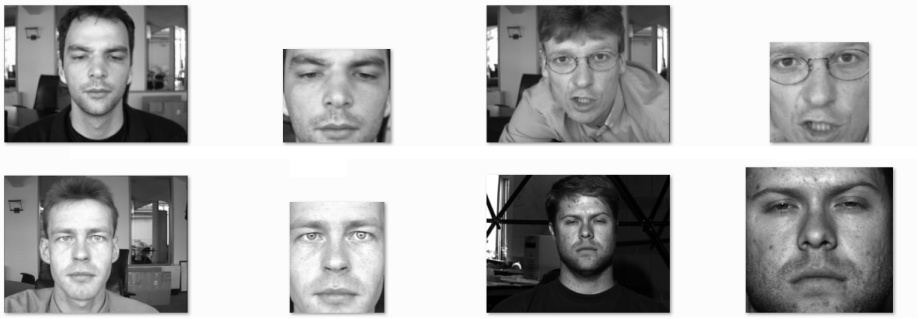


Figure 6.5: Example of cropped positive images.

Database B" consist in a dataset with a total 5850 grayscale images with the subjects in diverse positions and light variations. The Bio-ID Face Database dataset consists of 1521 frontal grayscale images. Each one shows the frontal view of a face of one out of 23 different test persons with various expressions.

Since adding different poses would not serve the scope and goals of this dissertation we decided to use exclusively frontal faces, in an attempt to simplify development and analysis. Because of this constraint, the total number of available positive examples is 2172. In order to build the ground truth file, the images have to be manually selected and cropped. In order to crop images and create positive samples a tool named image clipper was used [2]. These cropped images, see figure 6.5, are the objects that the Haar classifier attempts to discriminate from negative samples. After manually filtering out images that were too dark or although being frontal poses only part of the face was illuminated a total of 1905 positive examples, and corresponding cropped versions, was attained.

The negative dataset influences both the training time and test performance. The harder and bigger the negative examples are the more training cycles will consume, but performance will tend to be better than when using than simple and small ones. The images selected were gather from the web [3] and consist of a total of 3019 examples of landscapes, objects, drawings, among others. A sample can be viewed in figure 6.6.

In the conducted experiments we consider two dataset ratios and two supervisor modes, which yields four experiments. The two datasets can be described as follows:

- Balanced – consisting in 1905 positive and 1905 negative examples.
- Unbalanced – consisting in 500 positive and 1000 negative examples.

[2]Image clipper — `http://code.google.com/p/imageclipper/`

[3]Tutorial haartraining — `http://tutorial-haartraining.googlecode.com/svn/trunk/data/negatives/`

Figure 6.6: Example of cropped positive images.

The Unbalanced scheme was created based on the study of Lienhart [49] that suggests the ideal ratio for a good classification is around 1:2 ratio between positive and negative sets .

In what concerns supervision modes we consider the previously mentioned modes:

- Addneg – At the end of each evolutionary run, the supervisor it adds all individuals classified as positive by the internal classifier to the negative dataset;
- Classif — It uses an external classifier to test the instances classified as positives during the evolutionary run. If the external classifier does not detect a face, adds the "false positive" to the negative dataset.

The external classifier, as stated before, is the one implemented by the FDlib.

We planned to make 5 boosting iterations for each experiment. Unfortunately, due to lack of computational resources, it became impossible to go beyond the first iteration. The training procedure the models of the second iteration was run during more than 200 hours, for each model, without reaching completion. Training time grew exponentially as new instances were added.This increase in time may be explained by two possibilities: (i) the growth of negative examples to process; (ii) difficulties to correctly perform classification after adding the individuals generated in the EC runs.

6.4 Experimental Results

The results from the experiments are analyzed by evaluating classifiers of the initial iteration, and after one boosting iteration. In table 6.5 it is noticeable that, in the Balanced experiments, the Addneg model attained a better performance in comparison with the other schemes, in terms of average percentage of correctly classified examples during training. Its false alarm rate is lower than the one attained by the initial model. On the downside, it has the highest number of misses. Classif attained the lowest results with the exception of Hits and Misses, where Addneg registered the lowest values.

In the Unbalanced experiments, table 6.6, we verify that the initial model has the best overall result. Although it has the highest percentage of correctly

Table 6.5: Training performance when using the balanced training set.

	Positive				Negative				Average
Model	H	M	FA	%C	H	M	FA	%C	%C
Initial	1732	173	175	0.842	0	0	105	0.951	0.896
Addneg	1677	228	41	**0.861**	0	0	25	**0.987**	**0.924**
Classif	1698	207	159	0.8331	0	0	125	0.945	0.889

Table 6.6: Training performance when using the unbalanced training set

	Positive				Negative				Average
Model	H	M	FA	%C	H	M	FA	%C	%C
Initial	406	94	13	**0.788**	0	0	29	0.971	**0.880**
Addneg	371	129	7	0.732	0	0	26	0.976	0.854
Classif	390	110	21	0.752	0	0	21	**0.980**	0.866

classified examples and hit rates, it has more FP's than the Addneg model. The Classif performed better than the others in what concerns the classification of Negative examples.

Table 6.7 presents, the number of evolved individual that are classified as faces per iteration. The Balanced scheme has registered a growth in the number of individuals classified as faces from the initial iteration to the first iteration. On the opposite side, Unbalanced started with the highest values of all the schemes, but its first iterations, Addneg and Classif, fewer images classified as faces are evolved. Obviously, since these are results from a single run, no definitive conclusions can be drawn.

In figures 6.7 and 6.8 examples from the first iteration of images classified containing faces are presented, to illustrate the different patterns that emerge from the experiments.

As for the second iteration of the Balanced experiment, the imagery produced by the two models, Addneg and Classif, presented in figures 6.9 and 6.10, shares some similarities. The same can be said regarding the imagery evolved by both models in the first iteration of the Unbalanced experiment, which is presented in figures 6.11 and 6.12.

It's important to notice that even after one boosting run the EC engine was always able to find images that are misclassified not only by the internal classifier but also by the external one.

Table 6.7: Number of evolved images classified as faces by the internal classifier during the evolutionary run of the first iteration (initial) and during the evolutionary run of the second iteration using the Addneg and Classif models (Addneg and Cassif, respectively).

	Balanced	Unbalanced
Model	#Faces	#Faces
Initial	692	1324
Addneg	1081	584
Classif	1237	1007

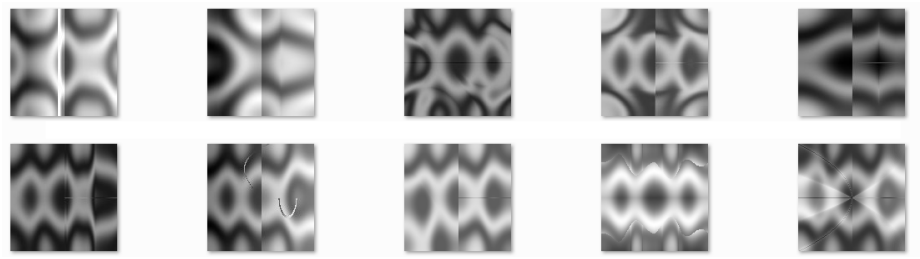


Figure 6.7: Samples of images evolved by NEvAr during the first iteration of the Balanced experiment and classified as faces by the internal classifier.

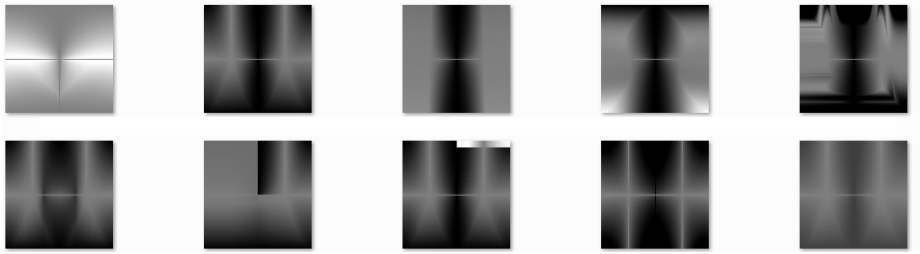


Figure 6.8: Samples of images evolved by NEvAr during the first iteration of the Unbalanced experiment and classified as faces by the internal classifier.

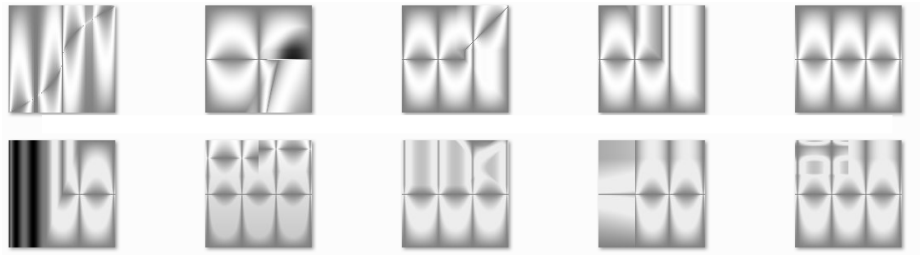


Figure 6.9: Samples of images evolved by NEvAr during the second iteration of the Balanced Addneg and classified as faces by the internal classifier.

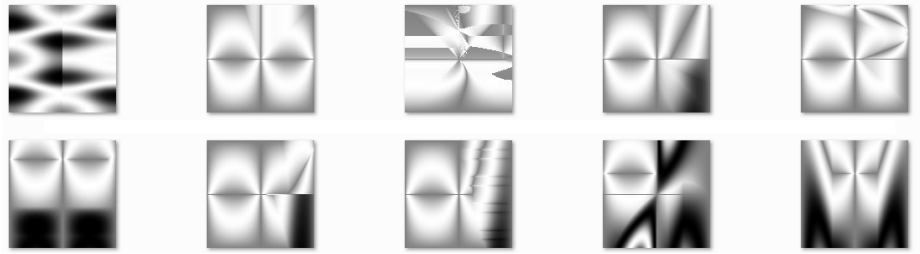


Figure 6.10: Samples of images evolved by NEvAr during the second iteration of the Balanced Classif and classified as faces by the internal classifier.

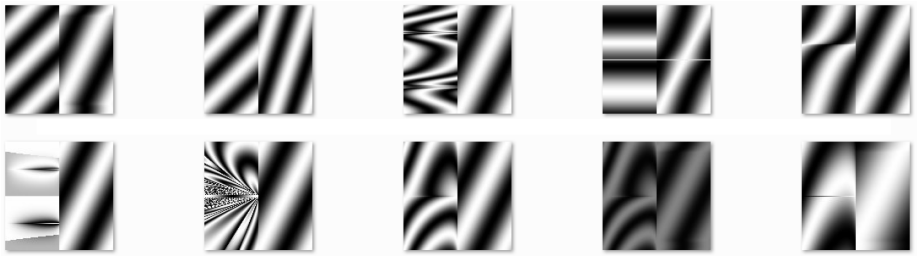


Figure 6.11: Samples of images evolved by NEvAr during the second iteration of the Unbalanced Addneg and classified as faces by the internal classifier.

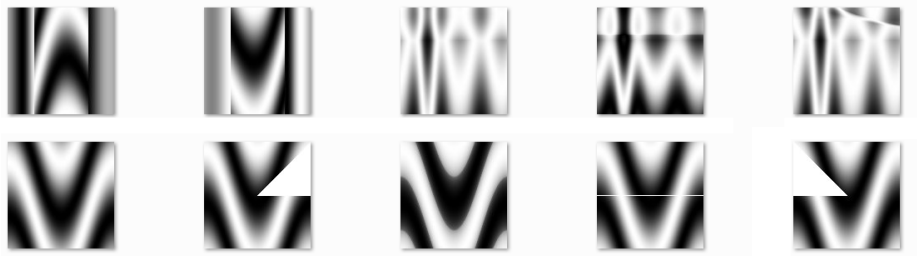


Figure 6.12: Samples of images evolved by NEvAr during the second iteration of the Unbalanced Classif and classified as faces by the internal classifier.

6.5 Assessing Classifier Performance

The results showed so far concern the performance in training in each of the experiments and iterations. Alone, this does not give a good assessment of classifier performance. In order to evaluate all the classifiers, internals and external, two independent validation sets were compiled:

- Feret – 902 images from the Facial Recognition Tecnology Database[4];
- Flickr images – 2166 negative images gathered from Flickr using the keyword "images";

Samples from the validation sets can be viewed in figures 6.13 and 6.14. The Feret validation set is composed by grayscale frontal faces, one face per image with a simple background. The images were manually selected and cropped. The purpose of using this validation dataset is to test the ability of the classifiers in detecting a clear frontal face. The Flickr image dataset consists in images retrieved from a search in Flickr and excluding from the resulting set, images that contain a frontal human face. These images consist in, landscapes, buildings, animals, computer screenshots, various objects, etc.

Table 6.8 presents the validation results attained in by the models of the balanced experiment. In what concerns the performance in the Feret set, Classif attains the highest performance with more Hits and less Misses than any other model and with a number of FA lower than the initial model. In the Flickr images dataset, Addneg attains the best overall performance, followed

[4]The Feret Database – `http://face.nist.gov/colorferet/colorferet.html`

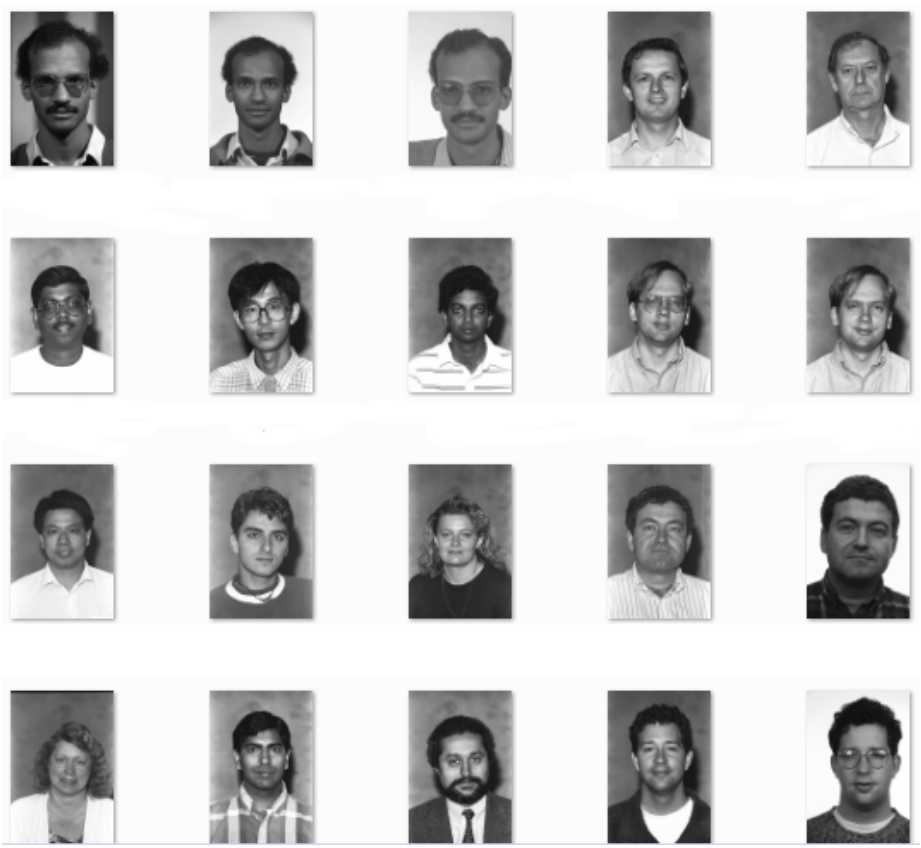


Figure 6.13: Feret dataset samples.

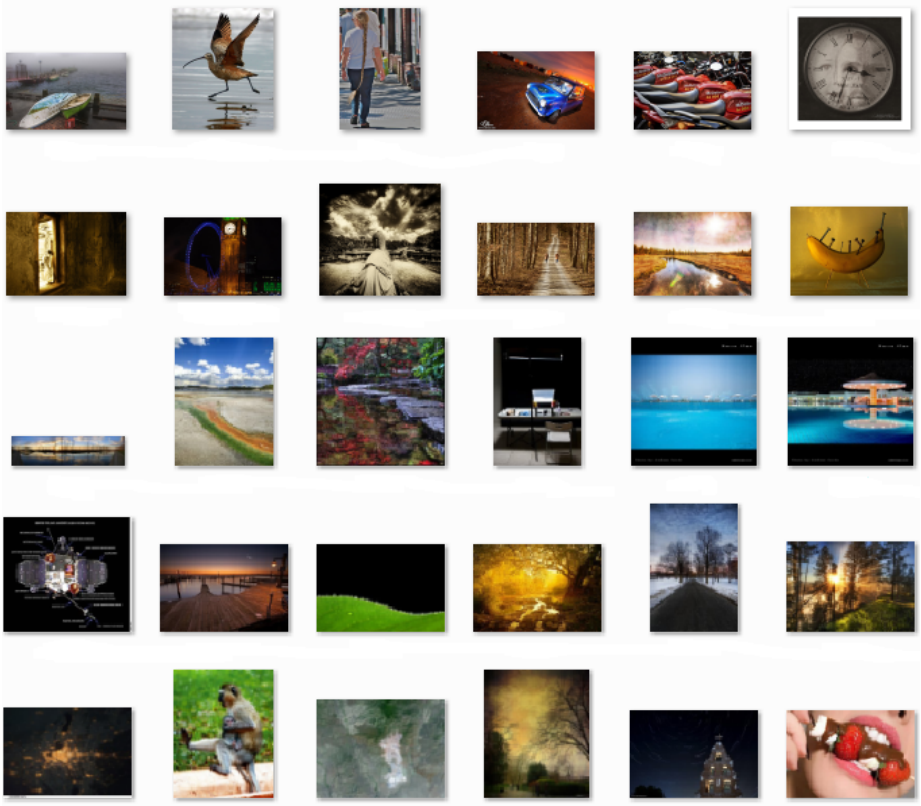


Figure 6.14: Flickr Images dataset samples.

Table 6.8: Balanced test performance

	Feret				Flickr Images				Average
Model	H	M	FA	%C	H	M	FA	%C	%C
Initial	749	153	26	0.807	0	0	297	0.890	0.849
Addneg	720	182	8	0.796	0	0	70	**0.970**	**0.883**
Classif	777	125	15	**0.850**	0	0	246	0.903	0.877

Table 6.9: Unbalanced test performance

	Feret				Flickr Images				Average
Model	H	M	FA	%C	H	M	FA	%C	%C
Initial	313	589	4	0.345	0	0	93	0.962	0.653
Addneg	265	637	13	0.288	0	0	83	**0.968**	0.628
Classif	499	403	18	**0.544**	0	0	89	0.961	**0.753**

by Classif. The number of FA generated by Addneg is significantly lower than the number FA generated by the other models. In summary, Classif attains better performance tha the initial model in both datasets, while Addneg attains a worse performance than the initial model on the Feret dataset (a difference of 1.1%) but significantly better performance on the Flicker validation dataset. In terms of average performance across the two considered validation datasets Addneg and Classif surpass the initial classifier.

In what concerns the Unbalaced experiment, whose results are presented in table 6.9, it is important to notice that the results attained in the Feret validation set are significantly lower than those attained by the Balanced experiment models. In spite of this, Classif attains the best performance on this dataset. In what concerns the Flickr images, all models attain high classification rates, with Addneg attaining the best performance for this set. Taking into account both datasets, the best performance is attained by Classif, and this is mainly achieved by the increase of performance in the Feret dataset.

The external classifier was tested for all the datasets available. Table 6.10 summarizes the results attained by this classifier. Surprisingly, FDlib only attains good results in the Feret dataset. In all other considered datasets it displays a performance that is worse than all considered models. Additionally, the performance on the Feret dataset is lower than those attained by all models of the Balanced experiment.

This indicates that it is possible to use a weak classifier as Supervisor and yet attain performance improvements. The evolutionary runs exploit shortcomings of the classifiers used to assign fitness, they are unable to identify and exploit the shortcomings of the Supervisor, since it isn't used to assign fitness. Thus, provided that the shortcomings of the supervisor are not the same as the shortcomings of the Classifier used to guide evolution, performance increases can be attained, as the experimental results indicate.

An example may help clarifying the above statement. Let's assume we have an internal classifier with a performance in the classification of negative instances of 99% and an supervisor with a performance of 90% and that the errors of these classifiers are unrelated. In each generation 100 individuals are created by the EC algorithm, meaning that the EC algorithm is likely to find a negative that is classified as a positive by the internal classifier in very few runs (the internal classifier misclassifies, on average 1 out of 100 negative individuals). Once this

Table 6.10: External classifier (FDlib) performance.

Data set	H	M	FA	%C	Average (%C)
Unbalance positive	346	154	44	0.634	0.634
Unbalance negative	0	0	538	0.675	0.675
Balance positive	1193	712	344	0.524	0.524
Balance negative	0	0	671	0.772	0.772
Feret	747	155	63	0.775	0.775
Flickr Images	0	0	1322	0.636	0.636

Figure 6.15: Frequently misclassified images of the Feret Image set.

false positive is found the EC will exploit this weakness and converge to similar images, which will be systematically misclassified by the internal classifier as FP. Quickly we will have hundreds of misclassified images. However, since the errors of the supervisor are not related with the errors of the internal classifier, the supervisor will only tend to fail on 10% of these images. Allowing us to identify them as FP and adding them to the training dataset, thus mending a shortcoming of the internal classifier.

Images of both validation datasets that are frequently misclassified by all models are presented in figures 6.16 and 6.15. As it can be observed the frequently misclassified positives appear to be characterized by bad lighting and tilted facial poses. In what concerns the negatives that are most frequently misclassified, it is harder to detect a common trend.

6.6 Summary

The conducted experiments indicate that the proposed boosting approach can be used to improve the performance of classifiers. In this particular case, the results suggest that it can be done in two ways:

- By adding individuals that were classified has positives by the classifier as negative examples
- By using an external classifier to filter what negatives should be really added.

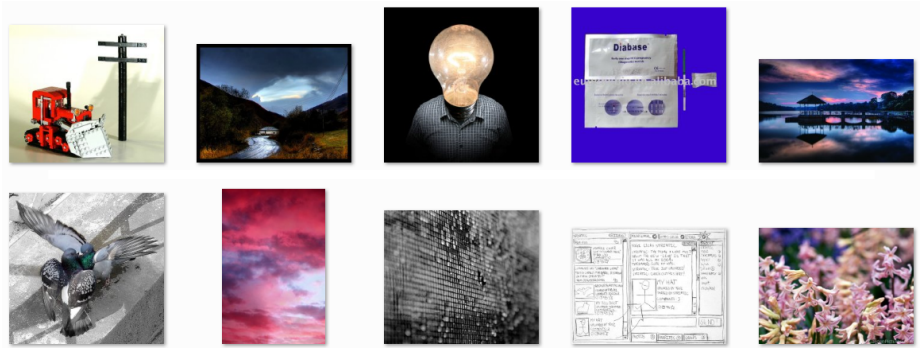


Figure 6.16: Frequently misclassified images of the Flickr image set.

Further study and experimentation will be required. The points of interest are mainly: (i) do more iterations (ii) understand further the cause of the exponential training time (iii) study other training and classifying parameters (iv) repeat the Classif experiment with other external classifiers.

Chapter 7

Conclusions and Future Work

In this dissertation we explored the use of EC to assess and improve classifier performance through the synthesis of new training instances. The approach relies on the ability of EAs to exploit weakness of the fitness assignment schemes used to guide evolution.

The proposed framework composed of three main modules – CS, EC engine, and Supervisor – with different roles:

CS The classifier system that is undergoing testing and improvement. Its outputs are used to guide the evolutionary engine;

EC The evolutionary engine is used to evolve instances that are misclassified by the CS;

Supervisor The module that determines which instances generated by the EC engine will be added to the new training sets;

The proposed framework is a boosting approach that promotes a competition between the EC engine and the CS, which across several boosting iterations is thought to lead to performance improvements.

In the course of this dissertation the framework was instantiated and tested in two image classification scenarios. The first scenario was inspired in the work of Machado et al.[59], and comprises the study of a boosting algorithm for a Style-Based Image classifier. A second scenario was tested by applying the framework to a Face Detection classifier.

The experimental results attained in these scenarios are presented and analyzed. Overall the results attained in the several validation tests performed indicate the viability of the proposed approach and its ability to improve the performance of the classifier in both settings.

The initial goal of this thesis was the development and refinement of a feature extractor for style-based image classification. However, the results attained in a preliminary experiment, conducted in the curse of this dissertation, which had the goal of evolving images evocative of human faces led us to shift the focus and goals.

It is important to notice, however, that the original goal was still accomplished. In the curse of this thesis several additions and refinements were made to the existent Feature Extractor. These changes are summarized in Chapter 5 and the feature extractor is described in A. These developments made possible the submission of a book chapter entitled "Computing Aesthetics with Image Judgement Systems" which has been recently accepted for publication as part of the book "Computers and Creativity", edited by Jon McCormack and Mark d'Inverno, to be published by Springer. [71].

It was not so far possible to publish the research described in the present document since the submission deadlines were not compatible with our research schedule and agenda. We have, nevertheless, plans to publish the presented work and continue research on this topic.

In terms of future work the most pressing issues are, in our opinion, (i) Performing a higher number of iterations, particularly for the experiments described in Chapter 6; (ii) Testing different supervisor modules, namely different classifiers and different image replacement/addition schemes; (iii) Performing additional validation experiments and analyzes to increase the reliability of the results and of their interpretation; (iv) Study alternative feature assignment schemes, in particular for the experiments described in chapter 5; (v) apply the framework to different problems, possibly outside the realm of computer vision tasks.

Appendix A

Feature Extractor

In this chapter the process of extraction of image characteristics that will be used in the system is covered and explained.

A feature or characteristic can be defined as a measurable property, physical or abstract, that is shared by all the entities of an entity category. Using this concept, it is possible to compare two works of art using just the palette of colors that exist in each one of them. These features are evaluated by means of metrics, quantifying each one of them. The development of metrics allows to identify a set of attributes that characterizes the object to be measured.

While in several previous works [69, 38, 90] metrics used are ad-hoc designed for a specific problem, this work uses metrics based in edge detection and in the complexity of the image. Some of these metric have been used previously by the group of investigation in other classification tasks, as the detection of one's author work of art or the resolution of psychological test of aesthetics [25].

For the realization of the described works a tool has been developed by Machado, a feature extractor, that from now on will be referred to as FE. This preliminary version contained some metrics for estimating of image complexity, implemented with the graphic library "IMAGEMAGICK"[1]. But there were some problems with portability and stability of the program, which led to different behaviours and results across platforms or by using different libraries versions.

For all the reasons mentioned, there was a need for a reconstruction of the FE, in an attempt to provide a stable and enhanced version to solve the pointed issues. The new solution was developed in C++ and using the Open Source Computer Vision library (OPENCV), a popular API for Computer Vision, multi-platform, portable, stable and with constant work, corrections and new features since its first release.[2]. This work has been made with the international collaboration of the *Artificial Neural Network and Adaptive System Lab from the University of A Coruña, Spain.*

For the next sections the FE system will be explained along with the theory behind the implemented algorithms and features. The image used to demonstrate most of the following algorithm was the *famous* Lenna image (fig A.1.

[1]ImageMagick: Convert, Edit, and Compose Images - http://www.imagemagick.org/script/index.php

[2]OpenCV (Open Source Computer Vision)- http://opencv.willowgarage.com/wiki/

Figure A.1: Lenna original.

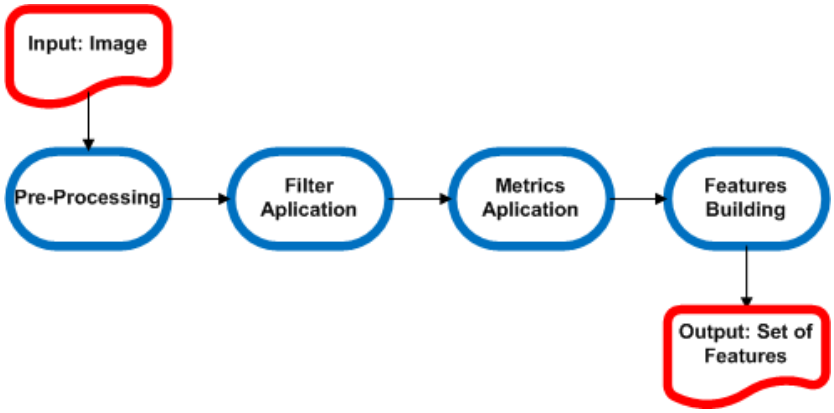


Figure A.2: Feature Extractor Overview.

A.1 Overview

The general design of the application, in terms of work flow, it is structured in (i) color channels, (ii) filters and (iii) metrics. This organization of the workflow allows to add a channel, or a filter with relative ease without affecting the design of the rest of the application. By means of configuration, each one of these elements can be activated or disabled independently through a configuration file, for instance, it is possible to turn off one of color channels available in the whole FE, excluding it from analysis and feature extraction. This configuration file allows the setup of each channel, filter, metric along with other parameters that the FE will use during it execution. This way, instead of calculating all the implemented features, it is possible to choose the features we want to extract, adapting to different context and situations. For instance, it is possible to skip the calculation of some available filters, reducing the program execution time greatly, gaining performance at the cost of skipping some unwanted or irrelevant features.

The feature extraction can be summarized to the following steps (fig. A.2):

1. Pre-processing the images, include all the transformation operations, partitioning and normalization to be applied to the input image.

2. Filters Application, is based on the application of different image filters and their respective variants to the resulting pre-processed images.

3. Metrics Application, the usage of certain methods based in measurements and statistical estimates of the filtered image.

4. Features Building, the extraction of the resulting metrics, with the purpose of obtaining the group of values characteristic of the input image.

A.2 Pre-Processing

Each input image, is loaded as a RGB color image (3 color channels), so that all the images have the same format at the entrance. As for the size, each image is resized to some dimensions of $width * height$ pixels, specified by the configuration file (default is $192 * 192$). A square size was selected because the most of the images have distinct width and height and for that reason, it was sought a neutral value for this matter in order to give the same size to all of the input images. We are aware that this adaptation in a ratio 1 to 1 between width and height may constitute a loss of information and, in the most cases, a deformation of the image, but in previous experiments and in other domains was proven that this transformation would not affect to the ability of the system to carry out the classification. This method as some advantage when confronting with other possible methods, for instance, there are some works which maintain the width and height relationship and use a background filler color to generate a square image, filling the image with irrelevant information, in contrast with the method employed, that attempts to maximize the relevant information. As for the chosen size, it was sought a compromise between the weight of the image, concerning the execution time, and the quality of the resized image. The size affects the calculation time of the extraction of each image metric almost exponentially. In previous experiments, the group used a size of $128 * 128$ but decided to increase this size to allow a higher quality of representation. Another constraint is that the size of each side should be a power of 2 or dividable by 4, because of the fractal compression algorithm. The value also needs to be divisible by 3, while maintaining the previous constrain because of one pre-processing operation named the rule of thirds, which will be explained later in this chapter. One advantage of this method is that by decreasing the image resolution, it will minimize the distortion caused by JPEG compression, present in many images in datasets used for experiments.

In addition to the analysis and exploration of the original image, it is also important to analyse its parts[59]. This partitioning process consists in segmenting the image in 5 equal parts: top left, top right, bottom left, bottom right, top left and center; store them and extract their features in the same way as the original image. In addition to this process a new region of the interest is extracted based on a classic rule of thumb in photography, the rule of thirds. This method has been used in similar works, revealing interesting features in the usage of this approach in image aesthetics [69].The rule of thirds is a way of describing where to place focal points in an image (fig. A.3).

Focal points are normally used to define areas of interest in an image. Theoretically this rule specifies that the main element, or the region of interest, in

Figure A.3: Application of the *rule of thirds* and highlighting of the focal points (in red).

an image should lie at one of the four focal points. This fact leads to the conclusion that a most of the main object of an image often lies on the periphery or even inside of the described region of interest. In order to obtain such region of interest, the image is divided in 9 equal sized regions and the center region is retained for later processing. The figure A.4 shows the process of image partitioning, including the rule of thirds, and the resulting images that can be used by the FE.

Regarding color, the image is be loaded as a RGB color image, 8-bit per channel, consequently the pixel values will be in range [0...255]. Afterwards the scaled image is converted to HSV color space (hue, value saturation). Each of these channels is stored as a individual grayscale image. From here on we will refer these auxiliary images as H, S and V. The complete picture, with the information of the three RGB channels, is stored and referred as ALL (from all channels) and is meant to be used only for the JPEG compression. By using the HSV color space model, it is important to take into account certain deficiencies which can cause random results when using this model. Figure A.5) shows a representation of HSV in a cone model. When pixel value of $V = 0$ the resulting color always black. In the case of $S = 0$ then the resulting color is always grayscale. Another problem occurs when $V = 255$ and $S = 0$, in which the color will be white. In any of these situations, it can not be assured that data from H will be correct as it depends directly from the transformation algorithm to HSV format. There are some ways to deal with these situations, the value of H is determined at random or assigned by a given constant. But this can be translated into noise when making a correct image classification using that information. In an attempt to address this deficiency it was created another grayscale channel, which is composed by a pixel by pixel multiplication of the S and V channels scaled to the range [0, 255]. From now on we will refer to it as CS, the colorfulness channel (see fig. A.5).

Jointly using the information of the H channel angle with CS we obtain a new value associated with the H channel in which the referred deficiencies are minimized. This also aims to increase the amount of information to be used by the classifiers themselves. The four auxiliary images obtained from the original, allow for a maximum of five different values associated with the channels used: ALL, S, V, H and the alternative of calculating H $_{CS}$. Also keep in mind that not all calculations will be made always but only in those cases where it is considered appropriate. In the same way that the calculation of ALL, consisting

Figure A.4: Resulting images from image partitioning and rule of thirds.

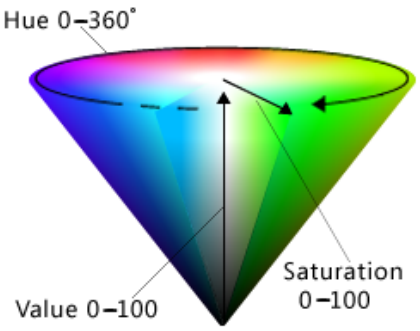


Figure A.5: HSV color space cone.

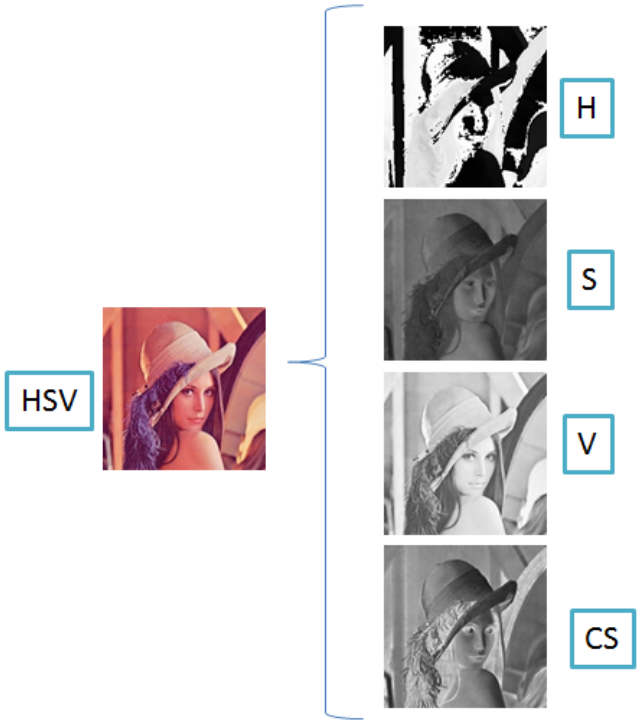


Figure A.6: The resulting color channels used in the FE.

of all channels, it is only used in the calculation of the JPEG compression, the calculation of H_{CS} is used only in the calculations of the Zipf law, pixels mean value and standard deviation.

In figure A.6, we can visualize the results of described pre-processing phase for the whole resized image. Note that every region of the image resulting from the partitioning phase is treated in the same way.

A.3 Image Filters

After pre-processing the image, each channel is treated as an individual image and each one is submitted to some image operations. These operations are referred as filters because its purpose is to select useful information from the whole image and use it for later analysis. With this objective in mind, the image operations included are constituted by edge detection operations, Sobel[79] and Canny[12], an image transform operation, the distance transform[1, 16], a quantization algorithm[28] and a salience algorithm, the subject salience[53, 46].

The edge detection algorithms, Canny and Sobel, were chosen due to the fact they are classic robust algorithms that have been used in similar works, as well in other areas[8, 77, 68, 19, 94, 33].

Distance transform algorithm is also a well known image processing algorithm, known by it's simplicity, adaptability and scalability and as also a wide range of applications[75, 15, 70, 9].

$$G = \sqrt{G_x^2 + G_y^2} \qquad \text{(A.1)} \qquad\qquad \Theta = \arctan(\frac{G_y}{G_x}) \qquad \text{(A.2)}$$

The above algorithms are included in the OpenCV library and only needed integration with the FE.

As for the quantization algorithm, it was integrated and adapted from giflib library [3], a Graphics Interchange Format (GIF) image format library. The GIF compression is one of the possible next possible features to be implemented in the FE and studied. Since it includes a quantization algorithm, it was found useful to use it as the FE quantization algorithm.

Lastly the salience algorithm chosen to be implemented was the subject salience algorithm. Although this algorithm was created to extract the subject of a given photograph, the reason to include it in the FE was to explore the concept of the image subject in any image as well study its usefulness in an image complexity analysis. This algorithm was totally implemented based on the Luo et al. work[53].

The following subsections describe these filters in detail.

Edge Detection

In terms of edge detection, the operators in use calculate the value of the first order derivatives in the horizontal (G_y) and vertical (G_x) directions. With these derivatives it is possible to calculate the gradient of the edges(equation A.1) and its direction (equation A.2).

More into detail, the Canny's algorithm[12] focuses on reducing noise by finding the gradient of image intensity. The designed method applies the following steps:

1. Filtering noise filter using gaussian smoothing or blur (filtering).
2. Find the most salient edge of the smoothed image by using the magnitude of the gradient.
3. Refining edges detected in the previous step using upper and lower thresholds that allow to constraint the magnitude of the intensities of the edges (hysteresis).

This filter, in the matters of the FE is applied vertically(fig.A.9b) and horizontally(fig.A.9a), by using G_x and G_y respectively.

The Sobel filter[79] is based on a discrete differentiation operator, computing an approximation to the slope of the function of image intensity at each point of it. The result of the Sobel operator is either the corresponding gradient vector or the norm of this vector. Mathematically, this operator uses two 3×3 matrices that are convoluted with the original image to calculate approximations of the derivatives - for horizontal and vertical variations. The Sobel filter calculates the gradient of the image intensity at each point, giving the direction of the greater variation from light to dark and the amount of variation in that same direction. This gives an idea of the variation of the brightness at each point, more smoothly or abruptly. With this filter it is estimated the presence of

[3] GifLib for Windows - http://gnuwin32.sourceforge.net/packages/giflib.htm

the light-dark transitions and what are it's orientations. With these light-dark variations corresponding to the intense and well-defined boundaries between objects, it is possible to obtain an edge detection.

Likewise the Canny filter, while concerning the FE, this filter is applied vertically and horizontally (fig.A.9e and fig.A.9d).

With the resulting images from horizontal(H) an vertical(V) edge detection, the an image with both edges can be computed in the following matter:

$$T_{xy_i} = \sqrt{(H_{xy_i})^2 + (V_{xy_i})^2} \tag{A.3}$$

Where H_{xy_i} and V_{xy_i} are the pixel intensity values of the ith-pixel of the horizontal and vertical edge detection results, respectively (fig.A.9f and fig.A.9c).

Distance Transform

The distance transform algorithm consists in a geometrical operator that maps each image pixel into its smallest distance to regions of interest. This algorithm is related to many other important entities such as medial axes, Voronoi diagrams, shortest-path computation, and image segmentation. In this case, the objective of the distance transform is to calculate the approximate or precise distance from every binary image pixel to the nearest zero pixel. With this concept in mind and with the help of the edge operator Canny, it is used to map the distance between edges and use it later for analysis. In order to achieve that objective the algorithm[9] is used, that is, for each pixel the algorithm finds the shortest path to the nearest zero pixel consisting of basic shifts: horizontal, vertical, diagonal or knight's move. The overall distance is calculated as a sum of these basic distances. Because the distance function should be symmetric, all of the horizontal and vertical shifts must have the same cost (that is denoted as a), all the diagonal shifts must have the same cost (denoted b), and all knight's moves must have the same cost (denoted c). In this case, the distance metric in use is the Euclidean distance. The parameters used by the FE are similar to the ones suggested in the original paper:

$$5 \times 5 mask, a = 1, b = 1.4, c = 2.1969 \tag{A.4}$$

By applying this 5×5 mask to all pixels of the image we get the distance transform of an image (fig.A.9g). Afterwards we scale the pixel intensity values based on local minimum an maximum to [0...255] and normalize the image, showing a distinct result from the step before (fig.A.9h). In this approach it is kept both images, the normalized and normal distance transform image, for later analysis.

GIF Quantization

The quantization method behind GIF compression algorithm is the median cut algorithm. This algorithm is characterized for being successful in identifying and preserving important colors in the color-reduced image, with little degradation towards the original. The median cut algorithm attempts to accomplish this goal by successively dividing the RGB color space into regions of equal color count, where the color count is the sum of the histogram counts enclosed by each of its region. The algorithm follows these steps:

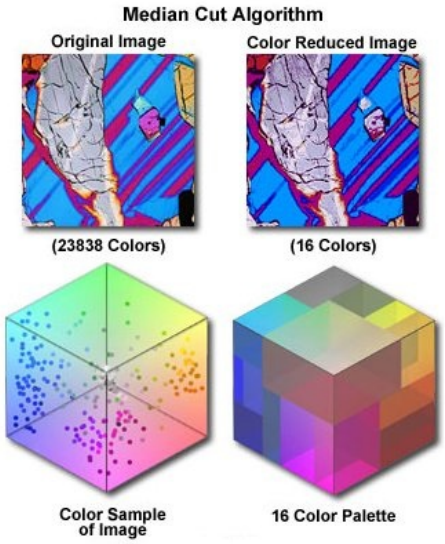


Figure A.7: An example of the median cut algorithm.

1. Enclose all of the colors in the image into a series of separate and smaller cubes defined within the boundaries of the RGB colorspace
2. For each sub-cube calculate the minimum and maximum value of the red, green, and blue components. The difference between these values is defined as the width of that color channel, and the widest of these determines the dominant dimension among red, green, and blue
3. The sub-cube with the largest dominant dimension is then determined, and that sub-cube is split into two smaller cubes along the median point.
4. This procedure divides the sub-cube into two halves so that equal numbers of colors are on each side, and the colors within the sub-cube are segregated into two groups depending on which half of the sub-cube they fall.
5. Repeat 2 until the desired number of sub-cubes has been generated.
6. Compute the color for each sub-cube by averaging all of the colors that are contained in the sub-cube.

The resulting set of colors constitutes the palette for the color-reduced image. Figure A.7 shows an example of the described process. An example of a quantization made by the FE is shown in figure A.9i.

Subject Saliency

Saliency can be defined by something that stands out relative to others, in the scope of Computer Vision, it is a point or a group of points that give more information then the surrounding ones. The saliency algorithm chosen to implement was the subject saliency algorithm also known as subject region

extraction[53]. In this case the saliency are defined by the points that most likely describe the subject of the picture.

In Luo et al. work, this algorithm was formulated based on an idea that the subject in a photograph would be more clear and the background would be blurred. Therefore the algorithm aims to extract the clear region of an image, which theoretically holds the subject and then extract it from the background.

This algorithm uses images statistics to detect 2D blurred regions in an image, based on a modification of Levin et al. work[46]. Next it maps the clear points of the image has a saliency map. With the saliency points calculated the following step is to enclose the subject in an area where most points are located.

The whole process is further described by these steps:

1. Define I as a matrix, holding the pixel intensity values of an input image.
2. Calculate vertical and horizontal first order derivatives. $I_x = I * d_x$ and $I_y = I * d_y$, where $d_x = [1, -1]$ and $d_y = d_x^T$
3. Create k kernels $k \times k$ with all coefficients $\frac{1}{k^2}$
4. Apply each kernel to I_x and I_y to obtain I_{xk} and I_{yk}
5. Calculate the pixel intensity histogram of I_{xk} and I_{yk} and obtain p_{xk} and p_{yk}
6. For each pixel in I_k, define the log-likelihood of the derivatives in its neighboring window $W_{(i,j)}$ with a size n. The log-likelihood can be defined by:

$$l_k(i,j) = \sum_{(i',j')\epsilon W_{(i,j)}} (\log(p_{xk}(I_x(i',j'))) + \log(p_{yk}(I_y(i',j')))) \tag{A.5}$$

7. The maximum value of $l_k(i,j)$ will define the kernel (k) that best describes the neighboring window statistics. If the best describing kernel of a pixel is equal to 1 then the pixel is in the clear area, otherwise is in the blurred area.
8. Compute the saliency map :

$$U(i,j) = \begin{cases} 1 & , best_k = 1 \\ 0 & , best_k > 1 \end{cases} \tag{A.6}$$

9. To enclose the subject, project U onto x and y axes independently. $U_x(i) = \sum_j U(i,j)$ and $U_y(i) = \sum_i U(i,j)$
10. Calculate x_1 and x_2 by adding each value of $U(i,j)$, in $[0, x_1]$ and $[x_2, N]$ until the total value reaches $\frac{1-\alpha}{2}$ of the total value of U_x, where N is the size of the image in x direction. The variable α is a constant used to control the total value needed to enclose the subject, the recommended value is 0.9. Do it similarly in the y direction. The subject region is defined by $Region = [x_1, x_2] \times [y_1, y_2]$. The background region is obtain by computing the pixels of $I \notin Region$.

Figure A.8: Saliency map.

A result of the saliency map is depicted in figure A.8. Good reference values for the number of kernels (k) and for the neighbouring window would be 30 and 3, respectively [46, 53]. The resulting filtered images are shown in figure A.9j and A.9k.

In figure A.9, we can see the results, for the value (V) channel, from this phase of the FE.

A.4 Metrics

As it was seen so far, the input image passes through a series of transformations, standardizing and normalizing it for further analysis. After that phase, different temporary images are retrieved from the information of the HSV color channels and a new temporary image is created (CS). On each of these images, in a second step of the whole process, different image filters were used, resulting in a few more temporary image, each one of them holding unique characteristics and information.

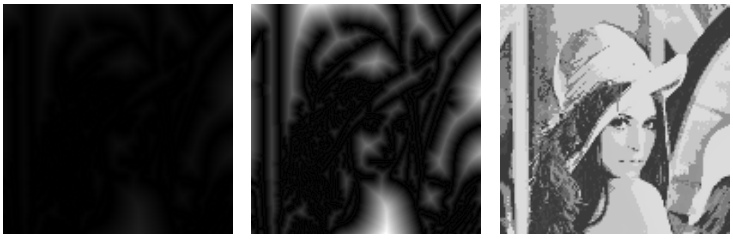
At this point, by using the resulting images, an application of image measurements, also referred as metrics, related with:

- Mean and standard deviation
- Complexity Estimates based on JPEG
- Estimates based on Fractal compression
- Frequency of currency Zipf (Zipf Rank-Frequency)
- Size frequency Zipf (Zipf Size-Frequency)
- Estimates of fractal dimension using box-counting

Within the metrics applied, it is also present the pixels mean and standard deviation of the image. These values are not related to measures of complexity, but are used later as a reference model to check the other estimators. These metrics are explained in more detail later on this section. Concerning the complexity estimates, there are three different families: (i) JPEG and Fractal Compression, (ii) Zipf's Law and (iii) Fractal Dimension.

(a) Canny Horizontal (b) Canny Vertical (c) Canny Horizontal and Vertical

(d) Sobel Horizontal (e) Canny Vertical (f) Canny Horizontal and Vertical



(g) Distance transform (h) Normalized distance transform (i) GIF Quantization

(j) Subject Saliency (k) Background Saliency

Figure A.9: Filter operations aplication.

Mean and Standard Deviation

The mean and standard deviation are simply calculated from the pixel intensity values of each image, with an exception for the channel H (Hue) from the image. Since Hue channel is circular, the mean and standard deviation are calculated based on the angle values of Hue and its norm. In addition, it is performed the multiplication of the Hue angle by the pixel intensity values of CS, and a new value of the norm is calculated using values from H and CS.

For the channel H it is considered the circular distance. The Hue channel is scaled to the range of [0...1] , and following formulas are used:

$$hx_i = \cos(H[x_i, y_i] \times 2\pi) \tag{A.7}$$

$$hy_i = \sin(H[x_i, y_i] \times 2\pi) \tag{A.8}$$

The above equations represent the Hue component of the ith pixel. Therefore, it is possible to calculate:

$$avg_{angle} = \begin{cases} \arccos(\overline{hx}) & , \quad \arcsin(\overline{hy}) >= 0 \\ 2\pi - \arccos(\overline{hx}) & , \quad \arcsin(\overline{hy}) < 0 \end{cases} \tag{A.9}$$

$$avg_{norm} = \sqrt{\overline{hx}^2 + \overline{hy}^2} \tag{A.10}$$

$$std = \frac{\sqrt{\sum_{i}^{s(I)} ((hx_i - \overline{hx})^2 + (hy_i - \overline{hy})^2)}}{s(I)} \tag{A.11}$$

The $\overline{hx}$ and $\overline{h}$ represent the mean values of the respective components, the image I and s is the function of file size.

The following changes in equations A.12 and A.13 allow the calculation of these metrics considering CS channel, generating the value it is referred to as H $_{CS}$:

$$hx_i = \cos(H[x_i, y_i] \times 2\pi) \times CS[x_i, y_i] \tag{A.12}$$

$$hy_i = \sin(H[x_i, y_i] \times 2\pi) \times CS[x_i, y_i] \tag{A.13}$$

These values conclude the calculation of the mean and standard deviation.

Complexity estimates

In terms of complexity estimates, as part of human perception, it is understandable that an image is transformed into an internal representation recognizable by the brain. This transformation would entail loss of information, causing this representation to be not exactly the same as the original image. According to Machado [56], the image aesthetic value can be determined according to their perceived complexity (hereafter CP) and complexity of the image (hereafter CI). According to this theory there should be a way to construct estimate values to both CP and CI complexities.

The fact that human beings do this pre-processing an image with the consequent loss of information serves as basis for using compression-based models,

which obtain a compact representation of an image and have the same deficiency. Machado used Fractal and JPEG compression to estimate CP and CI, respectively.

There is no evidence that the fractal compression constitutes a model of human perception. Nor are there any works that demonstrate a relationship between any of the mentioned compression methods and human perception of complexity of an image. However, these estimates have proven useful in image classification tasks according to stylistic criteria, such as, classifying an image according to the author of a painting or aesthetic [25]. Furthermore, these metrics have the advantage of being general and can be easily calculated.

A brief description of the chosen complexity estimates is showed and references to their relation with CP and CI. JPEG stands for Joint Photographic Experts Group. It is a standard method of compressing photographic images. The most common, viewed as default version of the JPEG is a lossy compression scheme. Although it exists a version of this compression that is lossless. By lossy it means that it looses information while compression the image, compression error. This is controlled by the compression ratio that can be assigned to perform the compression. This ratio affects the image quality and compression in an inversely proportional way, this means that, much higher ratio worsens the image quality but improves the compression value.

The JPEG implementation, makes use of the two-dimensional discrete cosine transform, DCT for short call. This transformation is based on Fourier transform (referred as DFT). Both the DFT and the DCT work through a series of finite numbers, but while the DCT only works with the cosines, DFT uses complex exponentials.

The whole JPEG compression is performed in four steps:

1. the original image is divided into different blocks of a certain size. Blocks normally used 8×8 pixels.

2. applies discrete cosine transform or DCT to each block.

3. Number of the coefficient matrix resulting from the implementation of DCT.

4. Coding of the image (Run Length Encoding) after the compression using the Huffman method. In fact, the format that holds a JPEG encoded stream is called JFIF[4], but it is usually called JPEG archive.

This algorithm is sensitive to changes in luminance (brightness changes) that become more noticeable in small areas with homogeneous brightness than in areas of greatest variation. This feature of the JPEG compression is also present in the human eye.

This compression method is essentially local, not taking into account any relationship between different elements of the image. The definition of complexity given by Moles [66] argues that aesthetic value is directly related to the complexity of the image. Proposes a measure of complexity based on the theory of information Shannon [74], the greater the predictability of the pixels, depending on the rest, the lower the complexity associated.

[4]JPEG File Interchange Format

Table A.1: Parameters used in fractal compression

	Low	**Average**	**Upper**
Image Size		256x256 pixels	
Minimum partition level	2	2	3
Maximum partition level	4	5	6
Maximum error per pixel	8	8	8

Within the relatively new methods in the field of image compression, are included those based on fractals and wavelets. Wavelet coding is based on the idea that the coefficients of a transformation that maps the pixels of an image can be encoded more efficiently than the original pixels. Barnsley [6] showed that any image can be represented through a set of fractal transformations so this kind of compression offer capabilities that generic representations similar of JPEG, does not.

Fractal compression method is based on the explanation of the similarities between different areas of an image (see figures A.10 and A.11). Thus, an image is encoded by a partitioning tree showing these similarities between different components. In particular we have chosen the tree model system TRNB or quadtree fractal compression, which used 4 levels of depth [17] with the set of parameters given in Table A.1.

This compression, according to Machado [56] is characterized by:

- The resulting image fractal compression seem to have superior quality to those obtained using JPEG compression, even when that fact is not verified mathematically.
- These images encoded using fractal compression have no fixed size and can be reproduced in any dimension. An image reproduced the original dimensions than have better quality than if it is produced by conventional methods of zoom.
- Explore the self-similarities present in an image is the idea that is based on fractal compression.

These characteristics of fractal compression allow the possibility to establish a logical resemblance to the way the human eye processes a picture in contrast to other compression methods.

The complexity estimate obtained by fractal compression is based on the same mathematical principles seen to JPEG compression, therefore expresses itself in terms of error and compression ratio.

The JPEG compression, that is a local method that ignores any relationship between the elements, it is used as an estimator for CP. On the other hand, in the case of fractal compression, attempts to find the relationships between different components of the image (such as self-similarity) in the same way that occurs in the human visual perception system. Because of this complexity will be used as an estimator for fractal CI.

As a practical example, when the complexity of an image increases the JPEG compression tends to produce worse results, which translates into increased

Figure A.10: Original Image

Figure A.11: Similarity

compression error estimator by increasing the value of CI. Moreover, as fractal compression does take into account the characteristics of the image and its possible relationship, it is reasonable to assume that its estimate for CP is higher than JPEG compression can provide.

It is not intended to suggest that the complexity of an image is equivalent to the estimator proposed as CI or that the human perception system specifically uses fractal compression, but both the complexity and the Fractal JPEG can be used as possible approximations for CI and CP, respectively.

We consider three levels of detail for metrics related to the JPEG and Fractal Compression: low, medium and high. For each level of detail, the process is the same, the image in analysis is encoded in a JPEG or Fractal format. Afterwards, the estimate of complexity is calculated using the following formula:

$$Complexity(I) = RMSE(I, CT(I)) \times \frac{s(TC(i))}{s(i)} \tag{A.14}$$

where $RMSE$ represents the root mean square error, CT is the JPEG or Fractal Compression rate of the image I and s is the file size function.

Zipf Law

Zipf's Law is the observation of phenomena generated by self-adaptive organisms, like humans, also known as the *principle of least effort*. Once the phenomenon has been selected for study, we examine the contribution of each case to the whole and rank according to their importance or predominance [97].

Bill Manaris has used the Zipf's Law as a criterion of beauty in music classification [62]. Since 2003, this researcher has been collaborating with the research group in musical classification, mainly using metrics based on Zipf's Law [61, 63, 59]. Based on the knowledge and experiences of the group, it were added several metrics based on this principle, applied to the pixel to pixel value in monochromatic channels.

In the work of the FE the metrics implemented regard the Zipf's law are: rank-frequency, size-frequency.

The calculation of the Zipf's rank-frequency is done by counting the number of occurrences of each value of a pixel in the range [0...255]. After these values are sorted in descending order and will be represented in a Cartesian axis according to their value and frequency. Then it is performed the linear regression of the

data, calculating the trend line. The metrics in used are the slope (m) and the linear correlation (R2) of the trendline.

In the case of the Hue channel, this metric is calculated in two ways. The first was described above. In the second, instead of only counting the number of occurrences of each value of Hue, we add also the channel values for the pixels corresponding CS (scaling back to [0...255]). The reason of this calculation is, the perception of tone depends on the saturation and the corresponding pixel value.

The metric Zipf size frequency is calculated similarly to the rank-frequency. Instead of using the value of each pixel, the difference between the value of a pixel with each of its neighbors is calculated. Next, count the total number of occurrences of differences in size from 1 to 255. Similar to the rank-frequency, the data is used to calculate the trend line and extract the slope and linear correlation.

In case of the Hue channel, a circular distance is considered. The Hue is scaled in the range [0...1], and the following formulas are used:

$$dx_i = \cos(H[x_i, y_i] \times 2\pi) \tag{A.15}$$

$$dy_i = \sin(H[x_i, y_i] \times 2\pi) \tag{A.16}$$

$$distx = dx_i - dx_{i+1} \tag{A.17}$$

$$disty = dx_i - dy_{i+1} \tag{A.18}$$

$$distance = \sqrt{distx^2 + disty^2} \tag{A.19}$$

$$circular_{distance} = \min(distance, |distance - \sqrt{8}|) \tag{A.20}$$

Since H values are between [0...1] , the maximum distance is achieved when $(distx = 2) \wedge (disty = 2)$ that results from $distance = \sqrt{2^2 + 2^2} = \sqrt{8}$. The equation A.20 ensures that we consider the minimum distance in the angle (eg. considering the distance clockwise and counterclockwise).

The size frequency of the hue (Hue) were also calculated using the channel CS. This is achieved by making the following changes in equations A.21 and A.22, respectively:

$$dx_i = \cos(H[x_i, y_i] \times 2\pi) \times CS[x_i, y_i] \tag{A.21}$$

$$dy_i = \sin(H[x_i, y_i] \times 2\pi) \times CS[x_i, y_i] \tag{A.22}$$

While calculating the value of the channel H through the channel CS, it is obtained a value for the hue channel variant H_{CS}.

Figure A.12: Lenna (V channel)

Figure A.13: Lenna Box-Counting

Fractal Dimension

Fractals have been invented by the mathematician Benoit Mandelbrot to describe the geometry of nature, whose complex and irregular forms beyond classical geometry. Mandelbrot has discovered that fractals are not just abstract objects because they exist in nature. This opened new areas of research in the chaos theory, especially regarding the differential equations chaotic behavior of their solutions.

Mandelbrot Fractal Dimension is mainly based on two principles:

- **Hausdorff-Besicovitch dimension** which generalizes the concept of the dimension of a real vector space. That is, the Hausdorff dimension of an n-dimensional vector space is equal to n. This means, for example, the Hausdorff dimension of a point is zero, the Hausdorff dimension of a line is one, and so on.
- **Hausdorff distance** measuring how far away from each other two compact subsets of a metric space. Hausdorff distance can be used to find a template of a given arbitrary image.

Fractal Dimension implies the existence of fractional measurements between integer dimensions. These values are approximated by appropriate measures in a progressively smaller scale. By representing a *log* vs *log* curve, results in a cartesian axis we can obtain a rough measure of the fractal dimension through its slope [64].

The last metric to be retrieved is the Fractal Dimension based on the method *Box-Counting* or counting boxes, chosen for its conceptual simplicity and ease of implementation. This method involves establishing a Cartesian coordinate system that contains the set of image points to be analyzed to which it will now be referred to as I. After, a count of $N_n(I)$ of squares (or cubes, or hypercubes depending on the size of the object to be analyzed) of size $\frac{1}{2^n}$ that intersect the total set.

The approach of the fractal dimension is given by the slope of the line obtained by linear regression of points $\log(2^n)$ $\log(N_n(I))$ [34].

The calculation of this metric is performed on monochrome images. Using a box size of $\frac{1}{4}$, the initial image is divided into quadrants. By using an escape operator condition ≥ 128, that is, if the box holds only pixels whose value is less than 128 it stops. On the other hand, if the condition occurs it is counted and

the box is re-divided to search recursively in each of the new partitions while the condition or size of box is not 1. A possible result could be observed in figures A.12 and A.13.

As before, once this is completed, a comparison between the size and number of occurrences is performed and both the slope and linear correlation of the trend line is calculated.

A.5 Feature's Building

After the process of metric calculation and understanding, the next step is to comprehend how they relate to each other in order to form the total number of values that are available at the end of the FE process.

From a given image it can be analyzed up to 7 parts of the image, the whole image and those obtained by partitioning the image, plus the rule of thirds. Each one them can generated up to 4 auxiliary images. These images are obtained from the color channels of the HSV model and CS. For each of these, and by combining up to 12 different filters, results in 336 auxiliary images. This set of images is used to calculate the described metrics: mean, standard deviation and estimates of complexity.

The mean and standard deviation generate 15 values from the channels H, S, V and variant H_{CS}. These include the H and H_{CS} average angle and norm.

The JPEG and Fractal compression has 3 values for each of the auxiliary image used, each serving a specific compression ratio (high, medium, low). JPEG is used for channels H, S, V and ALL, while for Fractal only use the first three. This is because only the metric related to the JPEG compression of all the proposed works with color images. These generate 21 metric values.

Metrics related to the Zipf's Law using the color channels H, S, V and variant H_{cs} from which has 2 different values, the slope of the trend line(m) and linear correlation(R2). In total there exists 16 values, 8 for the rank-frequency and 8 size-frequency.

Finally, the metrics of fractal dimension is calculated from the color channels H, S and V, but as happened with Zipf, it has 2 different values in response to the trend line and linear correlation, making a total of 8 values.

The metrics are applied to each auxiliary image, which totalizes the number of available features as $5,628$. Notice that in experiments with FE not all the features were used. Most experiments with the FE have been conducted with only 371, 603 or 804.

Appendix B

Populations

B.1 Style-Based Image Classification

B.2 Improving Face Detection

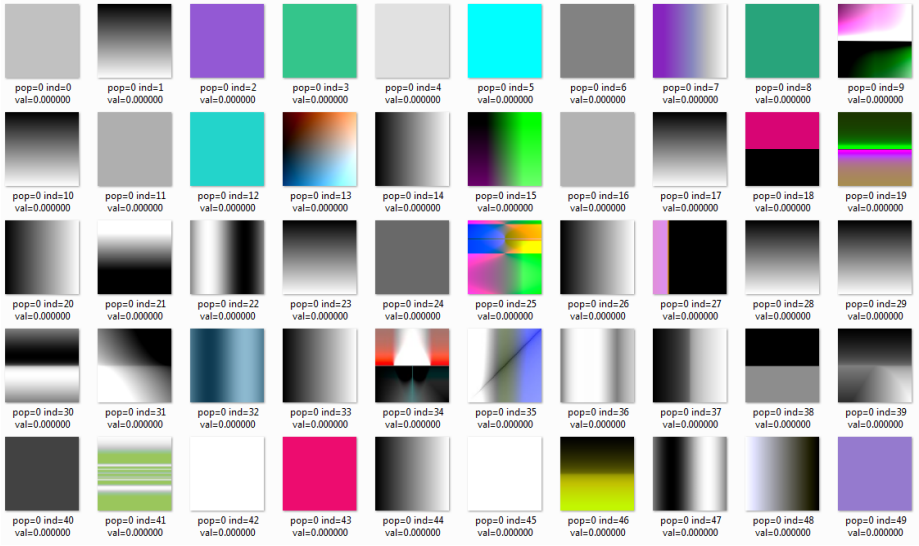


Figure B.1: NEvAr 2007 starting population

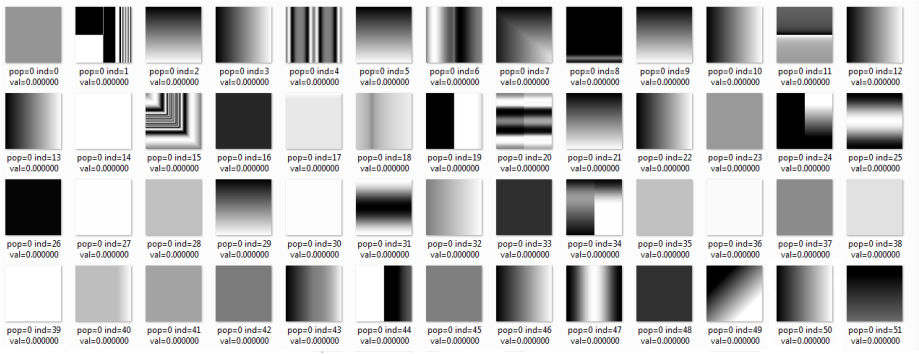


Figure B.2: NEvAr Faces starting population from 0 – 51.

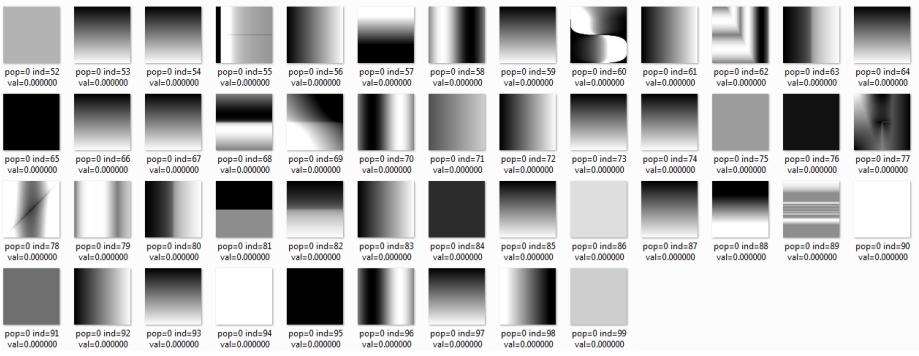


Figure B.3: NEvAr Faces starting population from 51 – 99.

Bibliography

[1] A., Rosenfeld and Pfaltz J.: *Distance functions on digital pictures.* Pattern Recognition, 1(1):33–61, 1968.

[2] Atkins, D L, R Klapaukh, W N Browne, and Mengjie Zhang: *Evolution of aesthetically pleasing images without human-in-the-loop.* In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8, 2010.

[3] Au, W H, K C C Chan, and X Yao: *A novel evolutionary data mining algorithm with application to churn prediction.* IEEE Transactions on Evolutionary Computation, (7):532–545, 2003.

[4] Ballard, D H and C M Brown: *Computer Vision.* Prentice Hall, 1982.

[5] Baluja, S, D Pomerlau, and J Todd: *Towards Automated Artificial Evolution for Computer-Generated Images.* Connection Science, 6(2):325–354, 1994.

[6] Barnsley, M F and A D Sloan: *A Better Way to Compress Images.* BYTE, January:215–223, 1988.

[7] Baro, X, S Escalera, J Vitria, O Pujol, and P Radeva: *Traffic Sign Recognition Using Evolutionary Adaboost Detection and Forest-ECOC Classification.* Intelligent Transportation Systems, IEEE Transactions on, 10(1):113–126, 2009, ISSN 1524-9050.

[8] Boehnen, Christopher Bensing: *IMPROVING 3D FACE RECOGNITION MODEL GENERATION AND BIOMETRICS*, 2009.

[9] Borgefors, Gunilla: *Distance transformations in digital images.* Comput. Vision Graph. Image Process., 34(3):344–371, 1986, ISSN 0734-189X.

[10] C, Darwin: *The Origin of Species*, 1859.

[11] Campos, T.
btxfnamespacelong E. de, B.
btxfnamespacelong R. Babu, and M Varma: *Character recognition in natural images.* In *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*, February 2009.

[12] Canny, J: *A computational approach to edge detection.* IEEE Trans. Pattern Anal. Mach. Intell., 8(6):679–698, 1986, ISSN 0162-8828.

[13] Cortes, Corinna and Vladimir Vapnik: *Support-vector networks.* Machine Learning, 20(3):273–297, 1995, ISSN 0885-6125.

[14] Dawkins, Richard: *The blind watchmaker: why the evidence of evolution reveals a universe without design.* W.W. Norton and Company, Inc., New York, 1987.

[15] Felzenszwalb, Pedro, David Mcallester, and Deva Ramanan: *A discriminatively trained, multiscale, deformable part model.* In *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR-2008*, 2008.

[16] Felzenszwalb, Pedro F and Daniel P Huttenlocher: *Distance Transforms of Sampled Functions.* Technical report, Cornell Computing and Information Science, September 2004.

[17] Fisher, Yuval (editor): *Fractal Image Compression: Theory and Application.* Springer Verlag, London, 1995, ISBN 0-387-94211-4.

[18] Fogel, D B and J W Atmar: *Comparing Genetic Operators with Gaussian Mutation.* In *in Simulated Evolutionary Processes Using Linear Systems, Biological Cybernetics*, pages 11111–11114, 1990.

[19] Freeman, William T and Edward H Adelson: *The Design and Use of Steerable Filters.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 13:891–906, 1991.

[20] Freund, Yoav and Robert E Schapire: *A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting*, 1995.

[21] Fukushima, K: *Cognitron: a self-organizing multilayered neural network.* Biological Cybernetics, (3-4):121–136.

[22] Gatarski, Richard: *Evolutionary Banners: An Experiment With Automated Advertising Design.* In *Conference on Telecommunications and Information Marketing*, Providence, Rhode Island, 1999.

[23] Georghiades, A S, P N Belhumeur, and D J Kriegman: *From few to many: illumination cone models for face recognition under variable lighting and pose.* IEEE Transactions on Pattern Analysis and Machine Intelligence, (6):643–660, ISSN 01628828.

[24] Graf J., Banzhaf W: *Interactive Evolution of Images.* 1995.

[25] Graves, Maitland: *Design Judgement Test.* The Psychological Corporation, New York, 1948.

[26] Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian Witten: *The WEKA data mining software.* ACM SIGKDD Explorations Newsletter, (1):10, ISSN 19310145.

[27] Hall, Mark A: *Correlation-based Feature Selection for Machine Learning.* Methodology, (April):17.

[28] Heckbert, Paul: *Color Image Quantization for Frame Buffer Display.* SIGGRAPH 82 Proceedings of the 9th annual conference on Computer graphics and interactive techniques, (3):297–307, ISSN 00978930.

[29] Hemert J., Eiben A: *Mondrian Art by Evolution.* 1999.

[30] Hjelmå s, E: *Face Detection: A Survey.* Computer Vision and Image Understanding, 83(3):236–274, 2001, ISSN 10773142.

[31] Holland, John H: *Adaptation in natural and artificial systems.* MIT Press, Cambridge, MA, USA, 1992, ISBN 0-262-58111-6.

[32] Holmes, G, A Donkin, and I H Witten: *WEKA: a machine learning workbench.* Proceedings of ANZIIS 94 Australian New Zealnd Intelligent Information Systems Conference, (3):357–361, ISSN 19305753.

[33] Huttenlocher, Daniel P, Gregory A Klanderman, Gregory A Kl, and William J Rucklidge: *Comparing Images Using the Hausdorff Distance.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 15:850–863, 1993.

[34] Huttenlocher, Daniel P, Gregory A Klanderman, Gregory A Kl, and William J Rucklidge: *Comparing Images Using the Hausdorff Distance.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 15:850–863, 1993.

[35] Ibrahim, Aladin M and Donald H House: *Genetic shaders: interactive and automatic shader generation.* In *ACM SIGGRAPH 97 Visual Proceedings: The art and interdisciplinary programs of SIGGRAPH '97*, page 189. ACM Press, 1997, ISBN 0-89791-921-1.

[36] Jesorsky, Oliver, Klaus J Kirchberg, and Robert W Frischholz: *Robust Face Detection Using the Hausdorff Distance.* Computer, (June):90–95.

[37] Jong, Kenneth A De: *Genetic Algorithms are NOT Funtion Optimizers*. Morgan Kaufman, Los Atlos, CA, 1993.

[38] Ke, Yan, Xiaoou Tang, and Feng Jing: *The Design of High-Level Features for Photo Quality Assessment.* Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, 1:419–426, 2006.

[39] Keane, and A. J. Keane, and S M Brown: *Thedesign Of Asatellite Boom Withenhanced Vibration Performance Usinggenetic Algorithmtechniques.*

[40] Kienzle, Wolf, G Bakir, and Matthias Franz: *Face detection-efficient and rank deficient.* Advances in Neural.

[41] Koza, John R and Riccardo Poli: *Genetic Programming.* In *Search Methodologies*, pages 127–164. Springer US, 2005.

[42] Krawiec, K, D Howard, and Mengjie Zhang: *Overview of Object Detection and Image Analysis by Means of Genetic Programming Techniques.* In *Frontiers in the Convergence of Bioscience and Information Technologies, 2007. FBIT 2007*, pages 779–784, 2007.

[43] Kriegman, D J, David J Kriegman, and Narendra Ahuja: *Detecting faces in images: a survey.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(1):34–58, 2002, ISSN 01628828.

[44] L J Fogel, A J Owens M J Walsh: *Artificial Inteligence Through Simulated Evolution*. Wiley, New York, CA, 1966.

[45] Langley, P, W Iba, and K Thompson: *An Analysis of Bayesian Classifiers*. In *Proceedings Of The National Conference On Artificial Intelligence*, number 415, pages 223–228. Citeseer, AAAI Press.

[46] Levin, Anat: *Blind Motion Deblurring Using Image Statistics*. Image Rochester NY, (3):841, ISSN 10495258.

[47] Lewis, Matthew: *Evolutionary Visual Art and Design*. In Romero, Juan and Penousal Machado (editors): *The Art of Artificial Evolution*, Natural Computing Series, pages 3–37. Springer Berlin Heidelberg, ISBN 978-3-540-72877-1.

[48] Li, C and T Chen: *Aesthetic Visual Quality Assessment of Paintings*. IEEE Journal of Selected Topics in Signal Processing, (2):236–252, ISSN 19324553.

[49] Lienhart, R, A Kuranov, and V Pisarevsky: *Empirical analysis of detection cascades of boosted classifiers for rapid object*. Lecture Notes in Computer, pages 297–304.

[50] Lienhart, Rainer, Alexander Kuranov, and Vadim Pisarevsky: *Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection*. Computing, (MRL):297–304.

[51] Lienhart, Rainer and Jochen Maydt: *An Extended Set of Haar-Like Features for Rapid Object Detection*. In *IEEE ICIP 2002*, pages 900–903, 2002.

[52] Llora, Xavier, Kumara Sastry, David E Goldberg, Abhimanyu Gupta, and Lalitha Lakshmi: *Combating user fatigue in iGAs: partial ordering, support vector machines, and synthetic fitness*. In *In Genetic and Evolutionary Computation Conference, GECCO 2005, Proceedings*, pages 25–29, 2005.

[53] Luo, Yiwen and Xiaoou Tang: *Photo and video quality evaluation: Focusing on the subject*. In *ECCV 08 Proceedings of the 10th European Conference on Computer Vision*, pages 386–399. Springer-Verlag, ISBN 9783540886891.

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

[55] Machado, P and A Cardoso: *NEvAr – System Overview*. In *Generative Art*, Milan, Italy, 2003.

[56] Machado, Penousal: *Inteligencia Artificial e Arte*. Phd Thesis. In Portuguese, 2007.

[57] Machado, Penousal and Amilcar Cardoso: *Generation and Evaluation of Artworks*. In *First European Workshop on Cognitive Modelling*, Berlin, 1996.

[58] Machado, Penousal and Amilcar Cardoso: *NEvAr - The Assessment of an Evolutionary Art Tool*. In Wiggins, Geraint (editor): *AISB'00 Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science*, Birmingham, UK, 2000.

[59] Machado, Penousal, Juan Romero, and Bill Manaris: *Experiments in Computational Aesthetics*. In *The Art of Artificial Evolution*. Springer, 2007.

[60] Machado, Penousal, Juan Romero, Bill Manaris, Antonino Santos, and Am\'ilcar Cardoso: *Power to the Critics - A Framework for the Development of Artificial Art Critics*. In *IJCAI'2003 Workshop on Creative Systems*, Acapulco, Mexico, 2003.

[61] Machado, Penousal, Juan Romero, Bill Manaris, Antonino Santos, and Amï¿½lcar Cardoso: *Power to the Critics - A Framework for the Development of Artificial Critics*. In *Proceedings of 3rd Workshop on Creative Systems, 18 th International Joint Conference on Artificial Intelligence (IJCAI 2003*, pages 55–64, 2003.

[62] Manaris, B, T Purewal, and C McCormick: *Progress Towards Recognizing and Classifying Beautiful Music with Computers - MIDI-Encoded Music and the Zipf-Mandelbrot Law*. In *Proceedings of the IEEE SoutheastCon 2002 Conference*, Columbia, 2002.

[63] Manaris, Bill, Penousal Machado, Clayton McCauley, Juan Romero, and Dwight Krehbiel: *Developing Fitness Functions for Pleasant Music: Zipf's Law and Interactive Evolution Systems*. In *Applications of Evolutionary Computing, EvoWorkshops2004: {EvoBIO}, {EvoCOMNET}, {EvoHOT}, {EvoIASP}, {EvoMUSART}, {EvoSTOC}*, LNCS, Lausanne, Switzerland, 2005. Springer Verlag.

[64] Mandelbrot, Benoit B: *The Fractal Geometry of Nature*. W. H. Freedman and Co., New York, 1983.

[65] Menezes, Telmo L T, Tiago R Baptista, and Ernesto J F Costa: *Towards Generation of Complex Game Worlds*. In *Computational Intelligence and Games, 2006 IEEE Symposium on*, pages 224–229, May 2006.

[66] Moles, Abraham: *Arte e Computador*. Republished by Afrontamento 1990, 1973.

[67] Papageorgiou, C P, M Oren, and T Poggio: *A general framework for object detection*. In *Computer Vision, 1998. Sixth International Conference on*, pages 555–562, January 1998.

[68] Perona, Pietro and Jitendra Malik: *Scale-space and edge detection using anisotropic diffusion*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12:629–639, 1990.

[69] Ritendra Datta Dhiraj Joshi, Jia Li and James Z Wang: *Studying Aesthetics in Photographic Images Using a Computational Approach*. Lecture Notes in Computer Science, 3953:288–301, 2006.

[70] Romdhani, Sami and Thomas Vetter: *Estimating 3d shape and texture using pixel intensity, edges, specular highlights, texture constraints and a prior*. In *Edges, Specular Highlights, Texture Constraints and a Prior, Proceedings of Computer Vision and Pattern Recognition*, pages 986–993, 2005.

[71] Romero, Juan, Penousal Machado, Adrian Carballal, and Jo btxfnamespacelong ao Correia: *Computing Aesthetics with Image Judgment Systems.* In McCormack, Jon and Mark D'Inverno (editors): *Computers and Creativity*, page to appear. Springer, 2011.

[72] Rowley, Henry A, Student Member, Shumeet Baluja, and Takeo Kanade: *Neural Network-Based Face Detection.* IEEE Transactions On Pattern Analysis and Machine intelligence, 20:23–38, 1998.

[73] Sha, Sha, Chen Jianer, Qin Ling, and Luo Sanding: *Evolutionary mechanism and implemention for recognition of objects in dynamic vision.* In *Computer Science Education, 2009. ICCSE '09. 4th International Conference on*, pages 178–182, 2009.

[74] Shannon, C E: *Prediction and entropy of printed english.* The Bell System Technical Journal, (30):50–64, 1951.

[75] Shotton, Jamie, Andrew Blake, and Roberto Cipolla: *Contour-based learning for object detection.* In *In Proceedings, International Conference on Computer Vision*, pages 503–510, 2005.

[76] Shyu, Brodley Kak, C Brodley, A Kak, C Shyu, J Dy, L Broderick, and A M Aisen: *Content-Based Retrieval from Medical Image Databases: A Synergy of Human Interaction, Machine Learning and Computer Vision.* In *Machine Learning and Computer Vision. In Proc. of the Sixteenth National Conference on Artificial Intelligence (AAAI'99*, pages 760–767, 1999.

[77] Siebel, Nils T, Sven Grï¿½newald, and Gerald Sommer: *Creating edge detectors by evolutionary reinforcement learning.* In *in Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2008), Hong Kong*, pages 3552–3559, 2008.

[78] Sims, K: *Artificial Evolution for Computer Graphics.* ACM Computer Graphics, 25:319–328, 1991.

[79] Sobel, I: *An isotropic 3 x 3 image gradient operator.* Machine Vision for Three-Dimensional Scenes, pages 376–379, 1990.

[80] Spears, William M, Kenneth A De Jong, Thomas Bäck, Thomas Ba, David B Fogel, and Hugo De Garis: *An Overview of Evolutionary Computation*, 1993.

[81] Spector, Lee and Adam Alpern: *Criticism, culture, and the automatic generation of artworks.* In *Proceedings of Twelfth National Conference on Artificial Intelligence*, pages 3–8. AAAI Press/MIT Press, Seattle, Washington, USA, 1994.

[82] Srinivasan, D and V Sharma: *Evolutionary computation and economic time series forecasting.* In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 188–195, 2007.

[83] Steffen, Jörg: *Text Classification using Weka.* Technology, 2011.

[84] Teller, A and M Veloso: *Algorithm evolution for face recognition: what makes a picture difficult.* In *Evolutionary Computation, 1995., IEEE International Conference on*, 1995.

[85] Tong, Hanghang, Mingjing Li, HongJiang Zhang, Jingrui He, and Changshui Zhang: *Classification of Digital Photos Taken by Photographers or Home Users.* In Aizawa, Kiyoharu, Yuichi Nakamura, and Shin'ichi Satoh (editors): *PCM (1)*, volume 3332 of *Lecture Notes in Computer Science*, pages 198–205. Springer, 2004, ISBN 3-540-23977-4.

[86] Viola, Paul and Michael Jones: *Rapid object detection using a boosted cascade of simple features.* pages 511–518, 2001.

[87] Wang, S L and A W C Liew: *Information-Based Color Feature Representation for Image Classification.* In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, 2007.

[88] Whitley, Darrell: *An overview of evolutionary algorithms: practical issues and common pitfalls.* Information and Software Technology, 43(14):817–831, 2001, ISSN 0950-5849.

[89] Wiens A.L., Ross B J: *Gentropy: Evolutionary 2D texture generation.* 2002.

[90] Wong, Lai kuan and Kok lim Low: *Saliency-Enhanced Image Aesthetics Class Prediction.* In *ICIP09*, pages 997–1000, 2009.

[91] Wong, Man Leung, Wai Lam, Kwong Sak Leung, and J C Y Cheng: *Applying evolutionary algorithms to discover knowledge from medical databases.* In *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on*, 1999.

[92] Wu, Yaowen, C Bauckhage, and C Thurau: *The Good, the Bad, and the Ugly: Predicting Aesthetic Image Labels.* In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 1586–1589, 2010.

[93] Xiao, Jing, Zbigniew Michalewicz, and Krzysztof Trojanowski: *Adaptive Evolutionary Planner/Navigator for Mobile Robots.* IEEE Transactions on Evolutionary Computation, page 1, 1997.

[94] Yang, Ming Hsuan, David J Kriegman, and Narendra Ahuja: *Detecting faces in images: A survey.* IEEE Transactions on pattern analysis and machine intelligence, 24(1):34–58, 2002.

[95] Zhang, C and Zhengyou Zhang: *A Survey of Recent Advances in Face Detection.* Learning, (June):1–17, 2010.

[96] Ziolko, Bartosz, Suresh Manandhar, Richard C Wilson, and Mariusz Ziolko: *Logitboost weka classifier speech segmentation.* 2008 IEEE International Conference on Multimedia and Expo, (3):1297–1300.

[97] Zipf, George K: *Human Behavior and the Principle of Least Effort*. 1949.