École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Département Informatique
5ième année

Final year project report

Image Quality Assessment : C++ Implementation and testing

Supervisors
Alireza Alaei
alireza.alaei@univ-tours.fr
Donatello Conte
donatello.conte@univ-tours.fr
Romain Raveaux
romain.raveaux@univ-tours.fr

Student
Maxime Martineau
maxime.martineau@etu.univ-tours.fr

DI5 2014 - 2015

Université François-Rabelais, Tours

generated on May 8, 2015

Contents

List of Figures

List of Tables

Thanks

I would like to address warm and vivid thanks to Alireza Alaei, Donatello Conte and Romain Raveaux for their help, patience and kindness.
I felt being part of a team more than a student during these 6 months and it was a wonderful feeling.
My thanks also go to people at Itesoft : Vincent Poulain d'Andecy, Ludovic Guillemot, Aurélie Joseph and the rest of the team who welcomed us during their *Innovation Days* workshop, helping research sprout in a brilliant way.

Introduction

This document is the report about Maxime Martineau's final year project about "Image Quality Assessment : C++ Implementation and testing".

This project was supervised by Alireza Alaei, Donatello Conte and Romain Raveaux from the LI Tours and will benefit to Itesoft, a company specialized in OCR, image-processing and document dematerialization.

From a presentation of the project follows a state of the art, the description of the proposed solution, some details about the delivered pieces of software, details about how the project was led and finally discussion about the global outcome.

Project presentation

1.1 Context

With the use of the computer tools, we are offered to store information.

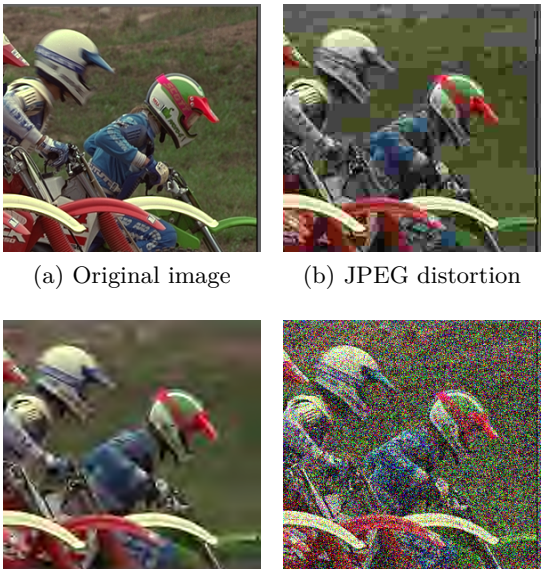
Given the growing amount of data, several issues are to be faced by information technology companies. Among these is the storing issue : the weight of all these document images files is expanding so quickly it becomes a critical point in terms of infrastructure and therefore in cost.

Moreover, facing the highly connected use of computer, the need of developing more efficient ways of transfering data is a direct consequence of the growing mass of data societies generate and have to deal with.

A part of the aforementioned data is stored as document images. These images use the same compression methods than any other image such as a scene image or a graphical image.

The question is the following : How can we reduce the size of these images without losing the underlying information ?

In order to significantly reduce the size of an image, we are likely to use lossy compression.



(a) Original image (b) JPEG distortion

(c) JPEG2000 distortion (d) White noise

Figure 1.1: Visual distortions

Lossy compression is a mean of reducing the space needed to encode data by approximating or simplifying the original data. Most of the methods are tunable in the sense that we are able to make the approximation more or less accurate and therefore more or less close to the original image.

Some examples can be seen on figure 1.1.

1.2 Goals

The compression evaluation will be specific to document images such as forms, letters or tables.
The goal is to provide a software that, given the image of a document, gives an indication on whether it should be more compressed or not — to keep the image readable —, and if it should, at which rate.
If it is already compressed enough, then the system should notice the user.

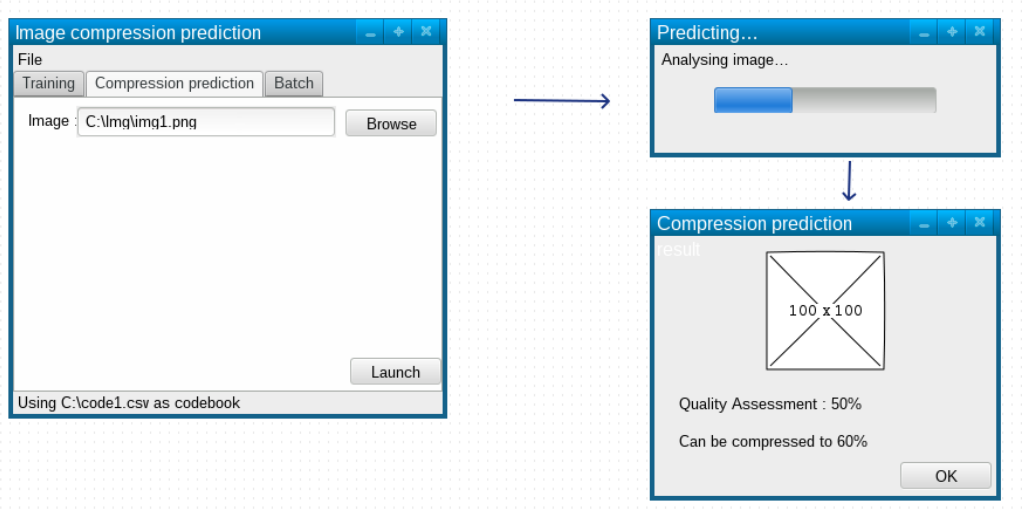


Figure 1.2: Sample view of the expected software.

It should have a "batch" functionnality which enables the user to assess and get results for a bulk of images.
Finally, to be extensible and easily integrable, the technical layer should be usable in separate programs as libraries.

1.3 Team

This project is part of a bigger scale project focused on Document Dematerialization which involves Itesoft and the *Laboratoire d'Informatique*, Tours — and more precisely the RFAI team — together.
Alireza Alaei, Donatello Conte and Romain Raveaux — who are part of the RFAI team in the *Laboratoire d'Informatique*, Tours — will supervise the project and its maintainer Maxime Martineau — Final Year Student in Computer Science in Polytech Tours.
Itesoft is an OCR, image-processing and document dematerialization company and will integrate and use the delivered libraries and the GUI in some of their final products. Ludovic Guillemot and Vincent Poulain d'Andecy are in charge of the project from Itesoft's side.

1.4 Provided work

The project is not about proposing a method to get compression prediction but studying and implementing a new one.

1.4.1 Approach to compression prediction

Alireza Alaei, Donatello Conte and Romain Raveaux came up with a new algorithm to solve this problem.
This method is based on an Image Quality Assessment method which is used to estimate the compression level of an image by correlating quality to compression level.
Some articles used for the method were provided as well.

1.4.2 IQA Matlab implementation

A Matlab/Octave implementation of this IQA does exist and was provided at the beginning of the project. This implementation will be used to both design and test the system. The expected accuracy and result are the one from this implementation.

1.4.3 MHOS

In order to establish a regression between quality and compression level, the following experiment has been led :
People have been asked to assess a bunch of document images in terms of quality and readability. These document images were generated and distorted at different levels and with different algorithms – some with JPEG2000, some other with classical JPEG.
To date, about twenty persons took this test. From their answers, a *Mean Human Opinion Score* — often mentioned as MHOS — for each image was deduced.
The system will be built on top of these elements.

1.4.4 Datasets

In order to test and evaluate the system, the following image sets are provided :

- The Berkeley Segmentation Dataset and Benchmark [5]
- The Computational Perception and Image Quality Lab image database [4]
- The LIVE Image Quality Assessment Database [7]
- The Tampere Image Database 2008 [6]
- Part of the DIQA database, provided and generated by Itesoft

These sets consist of several images. Some are considered reference images and distorted versions are provided.

1.5 Constraints

1.5.1 Use case

The software should be able to work on any type of compression. There is no *a priori* knowledge about the type of algorithm the user might use for their images.
As exposed before, the system is meant to work specifically on document images.

1.5.2 Technical constraints

The system is meant to work on Windows system and must be easily integrable. Therefore it must be implemented in Visual C++
The system must be lightweight and fast. OpenCV was then chosen to be used as an image processing library.

1.5.3 Results constraints

It has been chosen to use the existing Matlab/Octave implementation of the IQA library as an example. The system must provide at least as fast and accurate results as this implementation.

Presentation of the general problem

In this chapter are introduced the different research areas implied by the project.

2.1 Image Quality Assessment

Image Quality Assessment — as mentioned as IQA — is to provide a quantitative measure of an image's quality.

Besides other considerations, quality is here restricted to distortions such as blur, compression artifacts, noise and not about the content of the image itself — we could consider OCR recognition rate for document images or exposition for scene images for instance.

Here is a quick overview about this subject according to [2]

Approaches to provide this metric are either subjective or objective.

2.1.1 Subjective approaches

Subjective IQA approaches rely upon human assessments, either comparing images in pair — called pair-wise comparison — or using scales such as "Bad", "Poor", "Fair", "Good".

From all the assessments provided by the human subjects, a *Mean Human Opinon Score* (MHOS) is obtained for each image.

These methods require a significant amount of human subjects to assess all the sample images. Regarding this fact, these methods are costly and time-consuming.

2.1.2 Objective approaches

Objective IQA approaches only rely on the computer ability to assess images.

There are two main types of objective methods.

Full-Reference methods

The most obvious way for a computer to quantify the quality of an image is to have a reference. The reference is in fact a distortion-free version of the image (see fig 2.1)

(a) Reference image (b) Distorted image

Figure 2.1: Example of a reference-distorted image pair

Most of the methods do establish a similarity map — also called Local Quality Map, LQM — by giving a similarity metric per pixel. A global score is deduced from this map from statistical indexes such as min, max or average.

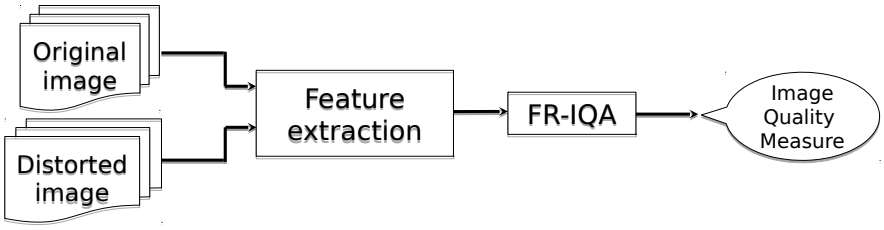


Figure 2.2: Full-Reference Image Quality Assessment flowchart [2]

These methods do need reference versions for every assessed image whereas, in most of the user scenario, it is not either convenient or possible to have a reference version.

No-Reference methods

No-Reference methods are not based on comparing two image versions. Some are distortion specific — they are conceived regarding one particular type of distortion to detect and quantify. That means one method is needed per distortion type.
Some other methods are distortion independent and can adapt to many types of visual distortions.

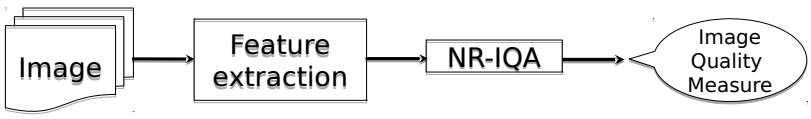


Figure 2.3: No-Reference Image Quality Assessment flowchart [2]

They are categorized as either **Natural Scene Statistics based** or **Training based**.

Natural Scene Statistics imply that natural scene images have inherent statistical characteristic that distortion might modify. The Natural Scene Statistics based approaches try then to detect these statistical variations in order to quantify the amount of distortion.
Training based approaches try to find features in images that characterize the distortions or lack of distortions. These features are associated to a quality measures during the training phase.

2.2 Compression estimation

Compression estimation consists in guessing how a given image has been compressed.
Research has been done on this topic but on specific compression types.

Proposed solution

3.1 Proposed compression prediction model

The approach lies on the assumption that the compression level of an image is correlated to its quality — we talk here about visual quality, as assessed from human.

In other words, if we have a visual quality measure that approaches the one given by human observators, then we can deduce the compression level applied to the image from this quality measure.

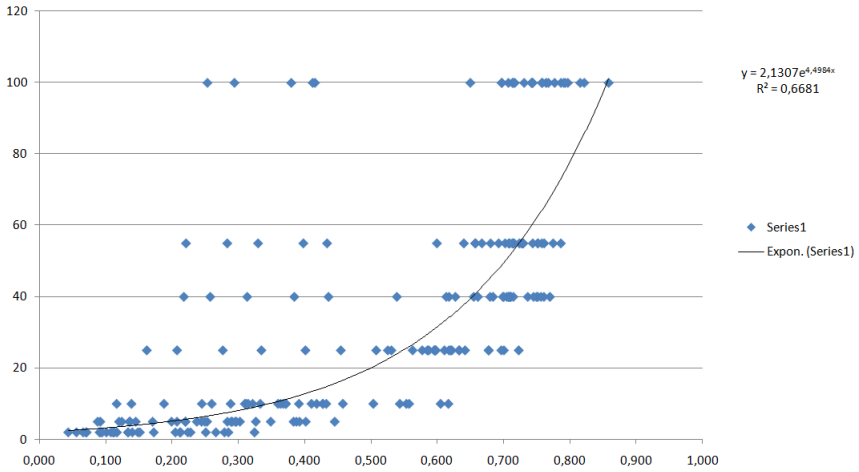


Figure 3.1: Establishing a correlation between quality and compression level (x is quality and y is compression level)

On the figure 3.1, we can see — to some extent — that the exponential (as plotted in black) is matching our MHOS points (as plotted in blue).

The x-axis is the quality whereas the y-axis corresponds to the compression level.

Of course, they are some erratic points called outlier due to the fact these points are human assessments. For instance, points on the top-left corner are clearly outliers because they are related to images that have been qualified as low-quality even though they are not compressed at all.

In the following paragraphs, the two steps to get the compression-level estimation are explained.

3.1.1 NR-IQA method

In this part are described the phases of the Image Quality Assessment method.

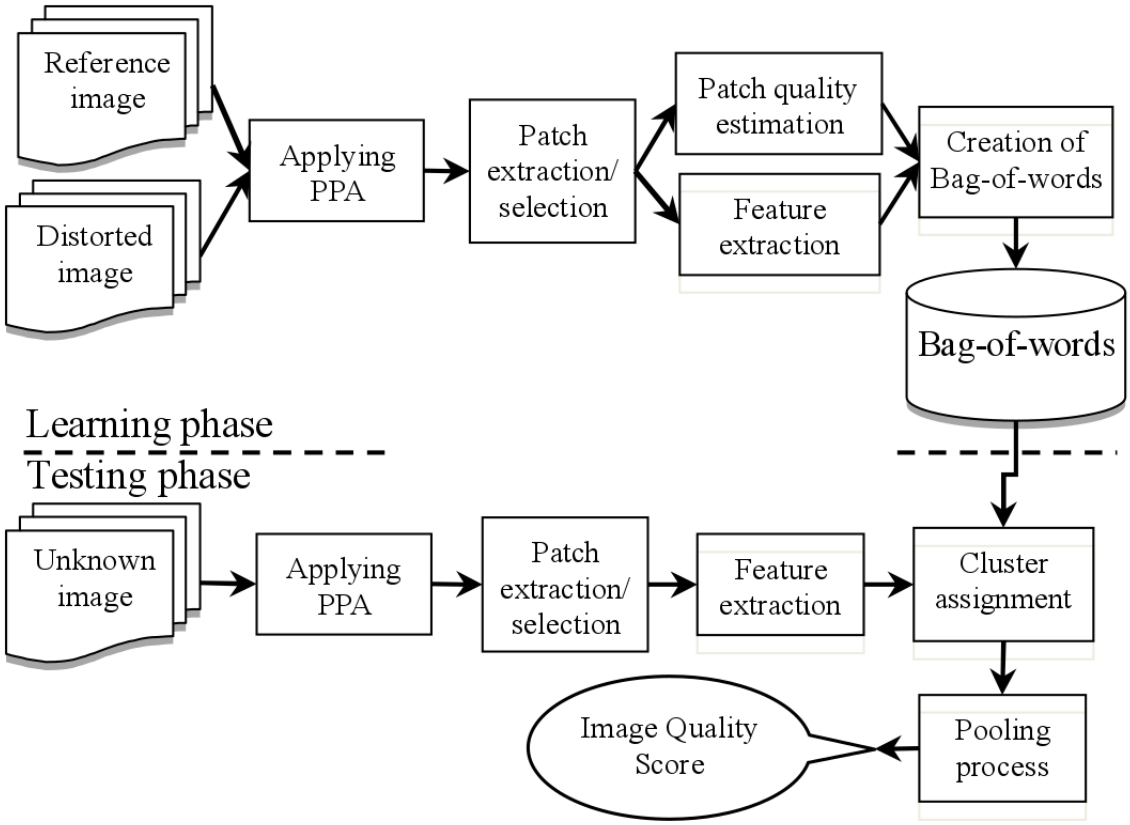


Figure 3.2: General block diagram of the IQA method [2]

As can be seen in the figure 3.2, the method has a training phase. During this training phase, the system learns the different distortions we need to recognize.

Learning phase

Given a set of images with reference and distorted versions, we want to build a database that will help recognizing the distortions. This database is called *visual codebook* or *bag of words*

Patch selection Given an image, the first step is to extract 8x8, 10x10, 12x12 or other dimension patches.

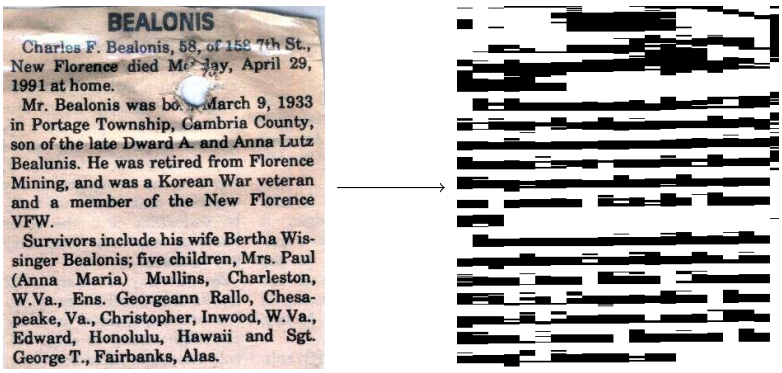


Figure 3.3: Binarizing the image

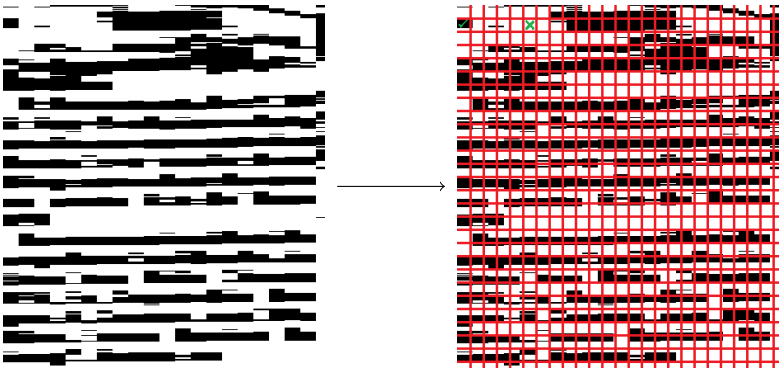


Figure 3.4: Extracting patches

The chosen criterion is to select patches that represent the foreground at most. In order to perform this choice, the image is binarized and patches are evaluated by the number of pixels belonging to the binarized foreground.

In figure 3.4 are shown the patches grid overlaid to the binarized image.

Given a patch P_k, let's note $|P_k|$ the total number of pixels of P_k and $|FG(P_k)|$ the number of *foreground pixels* within the patch.

If $T < \frac{|FG(P_k)|}{|P_k|} < 1$ then the patch should be selected (T is the threshold parameter) [1]

Extracting quality and features Once the patches are selected, two informations can be extracted out of them :

A quality measure is computed from a Full-Reference method. Both reference and distorted versions of the current image are indeed available.

Features are extracted from the distorted versions. It is in fact a feature vector (one coefficient per pixel.)

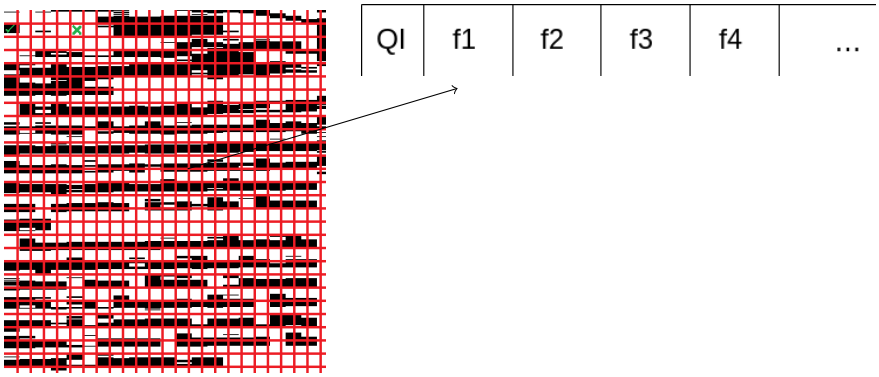


Figure 3.5: Performing FR-IQA (QI) and feature extraction for the image (fk)

At this point, a quality and a feature vector are kept for each patch.

Quality groups The feature vectors are grouped regarding the quality measure of the related patch.

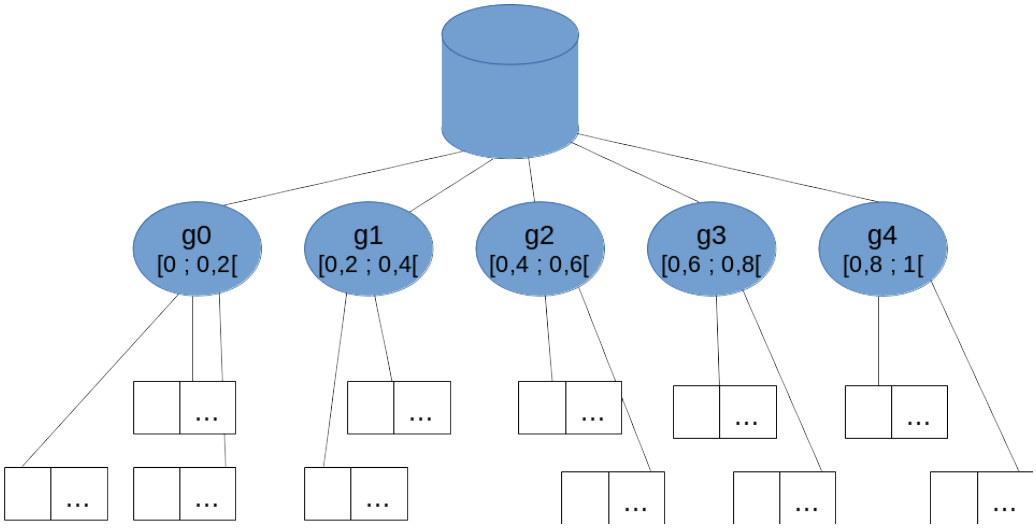


Figure 3.6: Grouping patches into groups

Most of the time, we worked with 10 groups dividing $[0; 1]$ in ten equal intervals.
One other important detail is that the quality measures must be normalized in a particular way in order to fill most of the groups. [8]

Clustering Once the feature vectors are quality-grouped, unsupervised clustering is performed on each group's vectors.
We want to obtain a given number of clusters in every group. In our experiments and testing, we worked with 30 clusters for each groups.
Another important thing is to normalize the clustering output data [8]

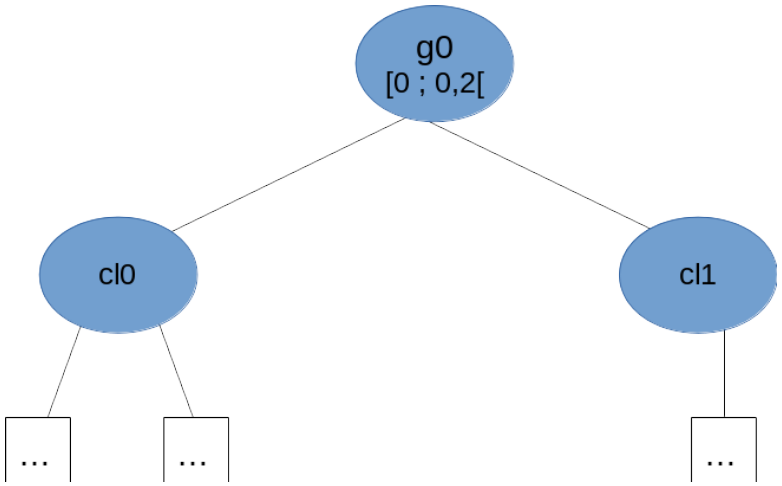


Figure 3.7: Clustering patches in the groups

What we obtain at the end of the learning phase are the quality-groups and for each group its clusters. These are the data used during the testing phase.

Testing phase

Given an unknown image, we want to obtain a quality measure for this image.

Getting patches As done in the learning phase, patches are extracted from the image following the same steps to select the patches.

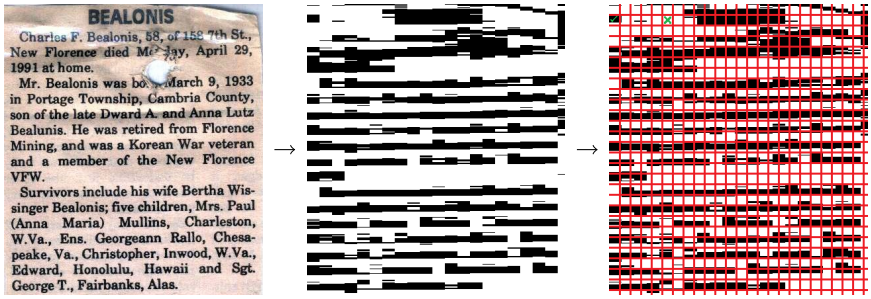


Figure 3.8: Getting patches

Extracting features Feature vectors are extracted for each selected patch. No need to notice that it is impossible to get a quality measure since it was done by using a Full-Reference method — which requires both the reference and distorted versions of the image.

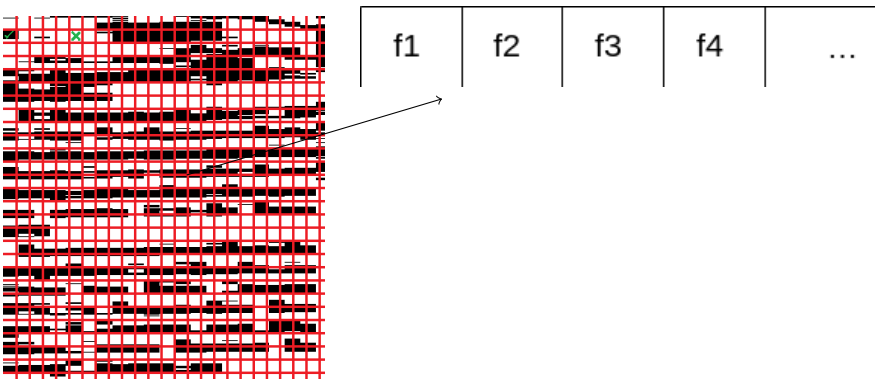


Figure 3.9: Extracting features from the image patches

Cluster assignments Given every extracted feature vectors, each of these are assigned to clusters — considering every cluster independently of their groups.

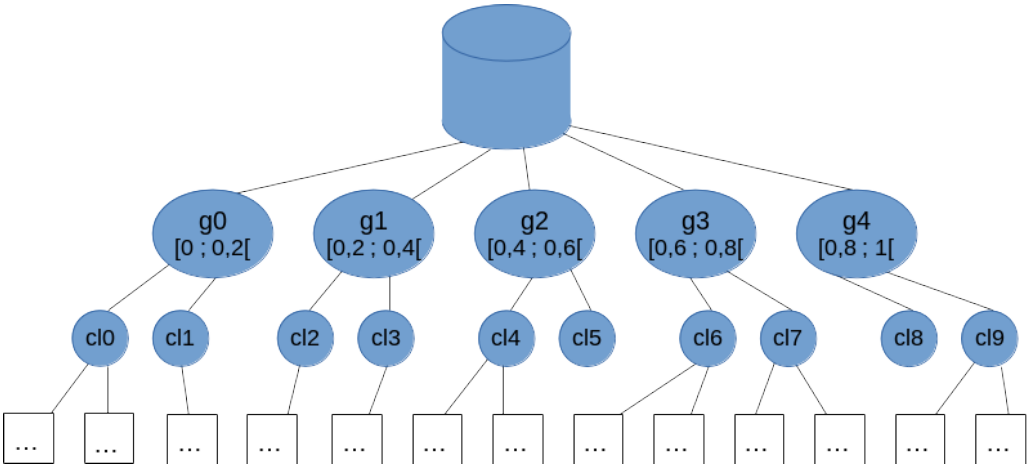


Figure 3.10: Assigning patches to clusters

A feature vector assigned to a cluster is assigned to the group of this cluster.

Pooling Number of assignments in every group are kept in a group assignment vector.

Given a group assignment vector $V = (n_1, \ldots, n_k)$ where k is the number of groups, the final quality score is the following :

$$IQA(V) = \frac{1}{N} \sum_{i=1}^{k} n_i Q(i)$$

Where $Q(i)$ is the quality score of the ith group and N is the number of patches.

3.1.2 Compression prediction

Now that the image has been given a quality measure, the compression-quality regression can be used to obtain the estimated compression-level for the image.

Estimating the compression level of an image

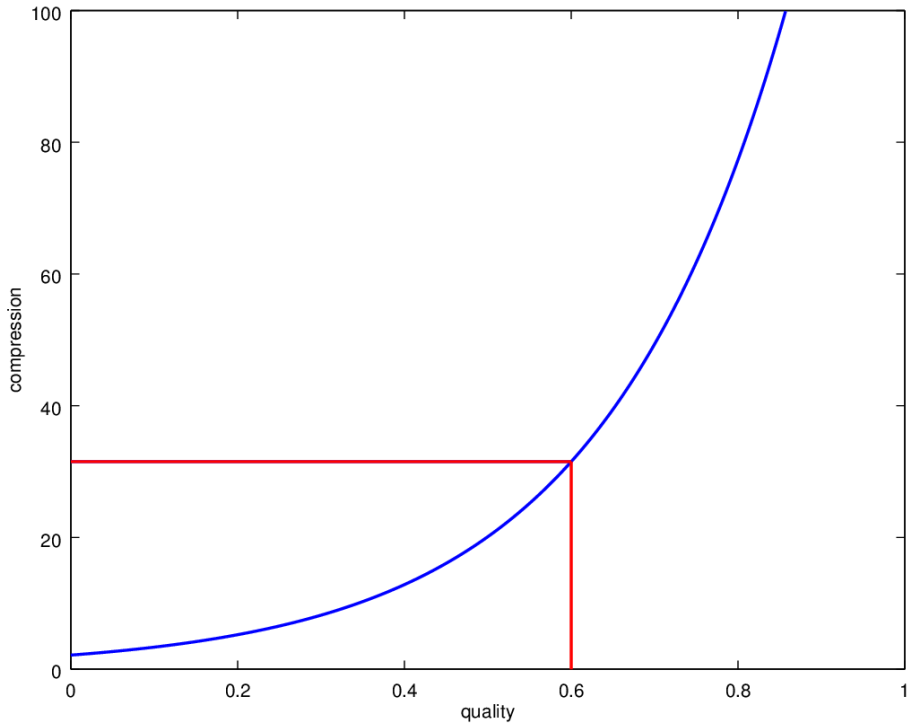


Figure 3.11: Using the regression to get compression level from quality

The figure 3.11 shows how to answer the question "How much is the image compressed ?"

Providing an advice on a potential stronger compression

But now, how to answer the question "How much more can the image be compressed ?"

In order to do so, we introduce a compression threshold $\tau \in [0;1]$ so that images compressed more than $1-\tau$ are considered illegible and unusable because too much compressed.
If the compression-level $c \in [0;1]$ for this image is so that $c > 1-\tau$, then the image cannot be compressed any more. Otherwise, the image can be compressed up to $c-\tau$

3.2 Algorithms used

The different steps of the algorithm can be done by different methods. However, a set of algorithms has been selected.

3.2.1 Patch extraction

In order to select the patches of the image that correspond to the foreground, a particular binarization algorithm that is called *Piece-wise Painting Algorithm* (PPA) [3]

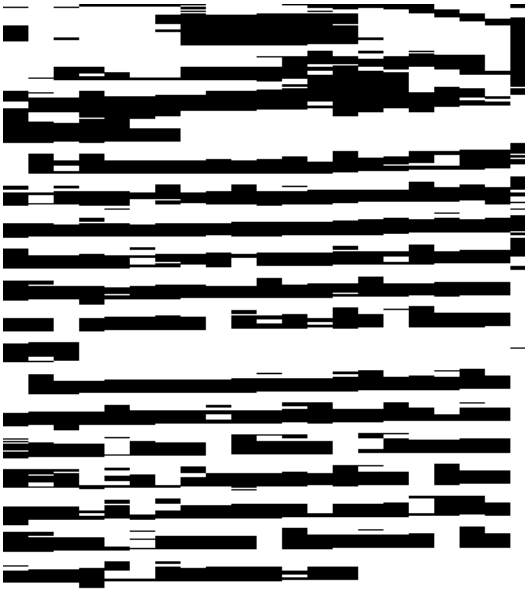


Figure 3.12: An image binarized with the Piece-wise Painting Algorithm [2]

PPA was fully implemented during this project.

3.2.2 Full-Reference Image Quality Assessment

The method chosen to get the quality measure during the training phase is called Feature Similarity (FSIM) [10]. It is a Local Quality Map based on two different features.
Phase Congruency is, according to [10], a feature that approaches the human vision system in the sense it describes how important a zone is in an image. It has been proven [10] that the most important parts of an image appear where the frequency-components of the Fourier space are congruent in phase.
Gradient Magnitude is the second feature used in FSIM that deals with contrast, Phase Congruency being contrast invariant.

3.2.3 Feature extraction

As a feature to use, Difference of Gaussian features were used at three different scales (or gaussian mask size) : 5x5, 13x13 and 25x25.

3.2.4 Clustering

In order to cluster the feature vectors into groups, the kmeans method was used with the euclidean distance.

3.3 Tools

In the following sections, the toolchain used to build the different pieces of software are shown.

3.3.1 OpenCV

OpenCV[1] — for *Open Computer Vision* — is a general purpose computer vision library written in C/C++. It provides many tools to manipulate image and video such as data structures and image processing algorithms implementations.

[1] http://opencv.org

As released under BSD license, not only it is allowed to be modified, redistributed and shared freely, but it is also free for personal, community *and professional use*.
It also provides several classifiers such as kmeans, neural networks, decision trees, SVMs, ...

3.3.2 Visual Studio/C++

Visual Studio/C++[2] is an Integrated Development Environment for C++. It consists of a complete C++ implementation.
The libraries computing IQAs and compression prediction are meant to be implemented in Visual C++.

3.3.3 STL

STL — for *Standard Template Library* — is a C++ library providing generic types and functions such as lists, arrays, queues, sorting algorithms or iterators.

3.3.4 C++/CLI

C++/CLI[3] is a C++ extention maintained by Microsoft enabling developers to use .Net framework components — usually available in C# — in Visual C++.

3.3.5 .NET framework

.NET is a framework from Microsoft providing ready-to-use libraries such as user interface, data handling, cryptography, ...

3.3.6 Windows Forms

Windows Forms is the user interface library built-in the .NET framework. It enables developers to quickly design and create well-integrated graphical user interfaces for Windows.

[2] http://msdn.microsoft.com/vstudio/
[3] Documentation for C++/CLI : https://msdn.microsoft.com/en-us/library/xey702bw.aspx

Programs details

This chapter presents the pieces of software that have been produced during the project.
These consist of two libraries and a graphical user interface.

4.1 NR-IQA Library

The No-Reference Image Quality Assessment library is the cornerstone of the project, regarding its complexity.
The figure 4.1 page 31 shows the class diagram for this library.

4.1.1 Usage

General API

The library should be simple to use for the following actions.

Training In order to train the system, the user can proceed in two different ways :

- One-shot by using `NRIQA` constructor providing `RefImgEntry` objects
- Step-by-step adding reference (`add_ref`) and distorted images (`add_dist`) and then calling `generate_codebook()`

```
// Step by step
NRIQA::CodebookParams p = NRIQA::CodebookParams::from_file("codebook_params.yml");
NRIQA::NRIQA nriqa(p); // Instance with parameters from file
// First reference image with its distorted versions
nriqa.add_ref("img/reference/1.png");
nriqa.add_dist("img/distorted/1_1.png");
nriqa.add_dist("img/distorted/1_2.png");
// Second
nriqa.add_ref("img/reference/2.png");
nriqa.add_dist("img/distorted/2_1.png");
nriqa.add_dist("img/distorted/2_2.png");
nriqa.add_dist("img/distorted/2_3.png");

nriqa.generate_codebook(); // Generating

// One-shot
vector<RefImgEntry> imgs;
// First reference image with its distorted versions
imgs.push_back(RefImgEntry("img/reference/1.png"));
imgs.back().add_distorted("img/distorted/1_1.png");
imgs.back().add_distorted("img/distorted/1_2.png");
// Second
imgs.push_back(RefImgEntry("img/reference/2.png"));
imgs.back().add_distorted("img/distorted/2_1.png");
imgs.back().add_distorted("img/distorted/2_2.png");
imgs.back().add_distorted("img/distorted/2_3.png");

// Generating codebook
NRIQA::CodebookParams p = NRIQA::CodebookParams::from_file("codebook_params.yml");
NRIQA::NRIQA nriqa(imgs, p);
```

Testing Once the user generated a codebook, everything is ready to assess images.

```
NRIQA::NRIQA n("codebook.yml");
double score = n.get_iqa("img.png");
std::cout << "Image score (" << path << ") : " << score << std::endl;
```

Configuration Configuration is done via a `CodebookParams` object.
Here is an example to create a parameter file :

```
NRIQA::CodebookParams c;
c.set_strategy("feature", "dog");
c.set_strategy("quality", "fsim");
c.set_strategy("patch_extraction", "ppa");
c.set_strategy("clustering", "kmeans");
c.set("patch_extraction.number_of_stripes", "40");
c.set("patch_extraction.overlap", "0");
c.set("patch_extraction.threshold", "0.6");
c.set("patch_extraction.patch_size", "8");
c.set("clustering.nb_clusters", "30");
c.set("clustering.epsilon", "0.001");
c.set("clustering.attempts", "300");
c.set("grouping.nb_groups", "10");
c.write("codebook_params.yml");
```

Flexibility

Given the general framework (see fig. 3.2), we can notice that the steps of the process can be performed by many different methods or algorithms.
At this point, we aren't sure of the optimal set of used methods. That is why the system must be easily modified and revamped in the sense that we must be able to change one method by another as handily as possible.

4.1.2 Modules presentation

NRIQA

`NRIQA` is the main object library users might have to manipulate. It contains all the methods needed to use the library in a third-party software.

Codebook

The `Codebook` object represents the visual codebook or bag of worlds, containing groups and clusters and performing assignment to it.

CodebookParams

`CodebookParams` is the structure responsible of both parameters of the algorithm and chosen methods/strategies.

ImgStruct

`ImgStruct` corresponds an image. It represents a reference version and optionally a distorted one.

RefImgEntry

`RefImgEntry` is useful to pass the images used in the learning phase.
A `RefImgEntry` object consists of a reference image and of its distorted versions.

Patch

A `Patch` corresponds to a part of an image, with references to its reference and distorted versions.

Group

A `Group` contains feature vectors — before clustering — or clusters. It has a quality.

Cluster

A `Cluster` object contains one or many feature vectors.

ComputingCache

A `ComputingCache` is an object able to store objects such as intermediate matrices in processes.
For instance, the FR-IQA method does need to compute features on full image instead of on patch only. Since quality is computed on each patch, we don't want to compute this full-image feature every time but only once and store it afterwards.

Strategy

The `Strategy` module contains every interfaces and classes needed for the strategy pattern to work.
Steps such as patch extraction or feature extraction can be done using many different algorithms. That is why, instead of putting the code directly in the classes, it is put in outter classes that are called strategies.

Tools

The `Tools` module contains the main helper needed for the different algorithms to work that are not provided by OpenCV.

4.1.3 Algorithms implementations

Every image processing algorithm did needed custom implementation at the exception of the kmeans (provided by OpenCV).

4.2 Compression Prediction

Compared to the NR-IQA library, the compression prediction library is pretty minimalistic. It consists of a simple function mapping and of thresholding.
It also enables the user to save the parameters in a simple text file.

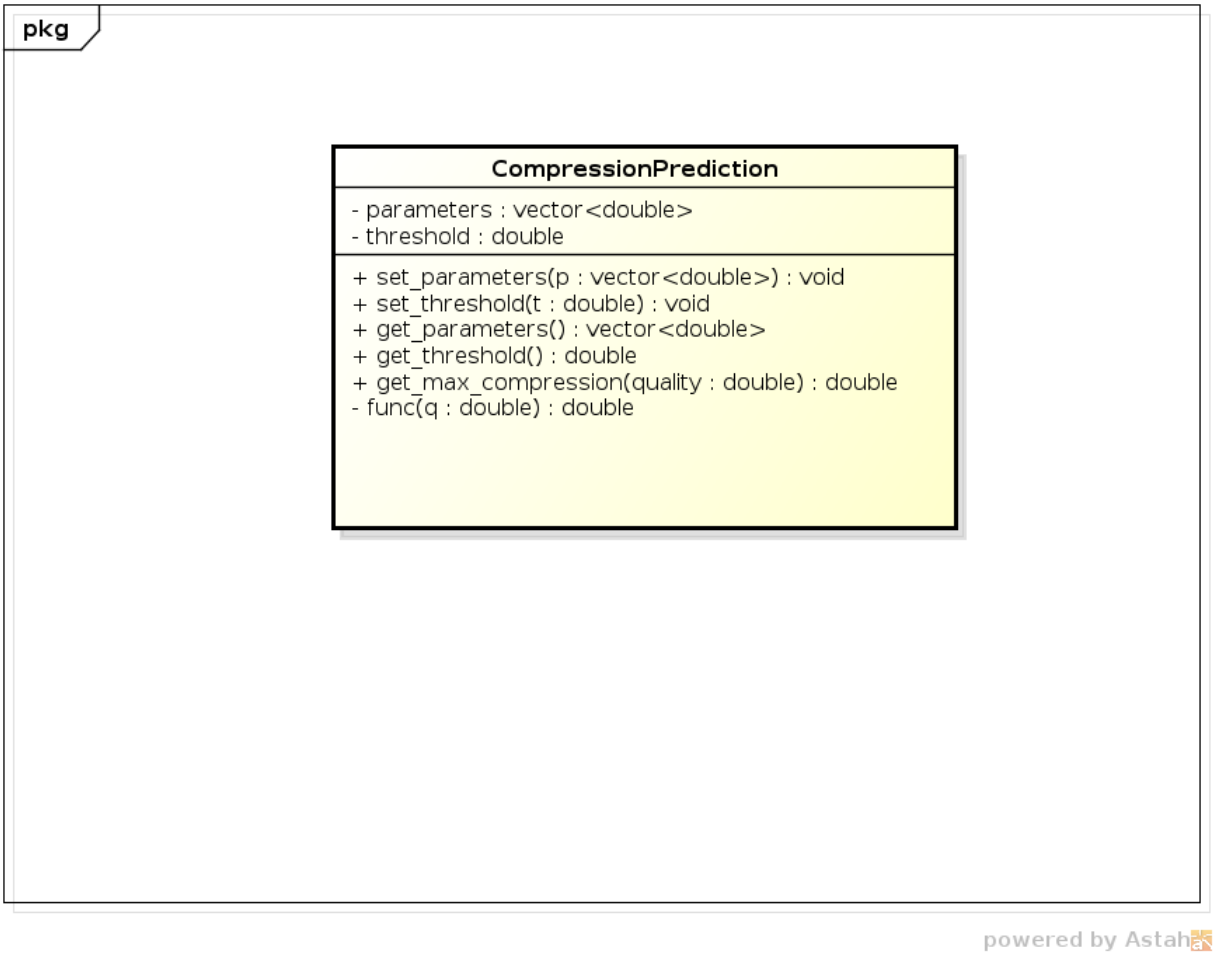


Figure 4.2: The Compression Prediction library class diagram

4.3 GUI

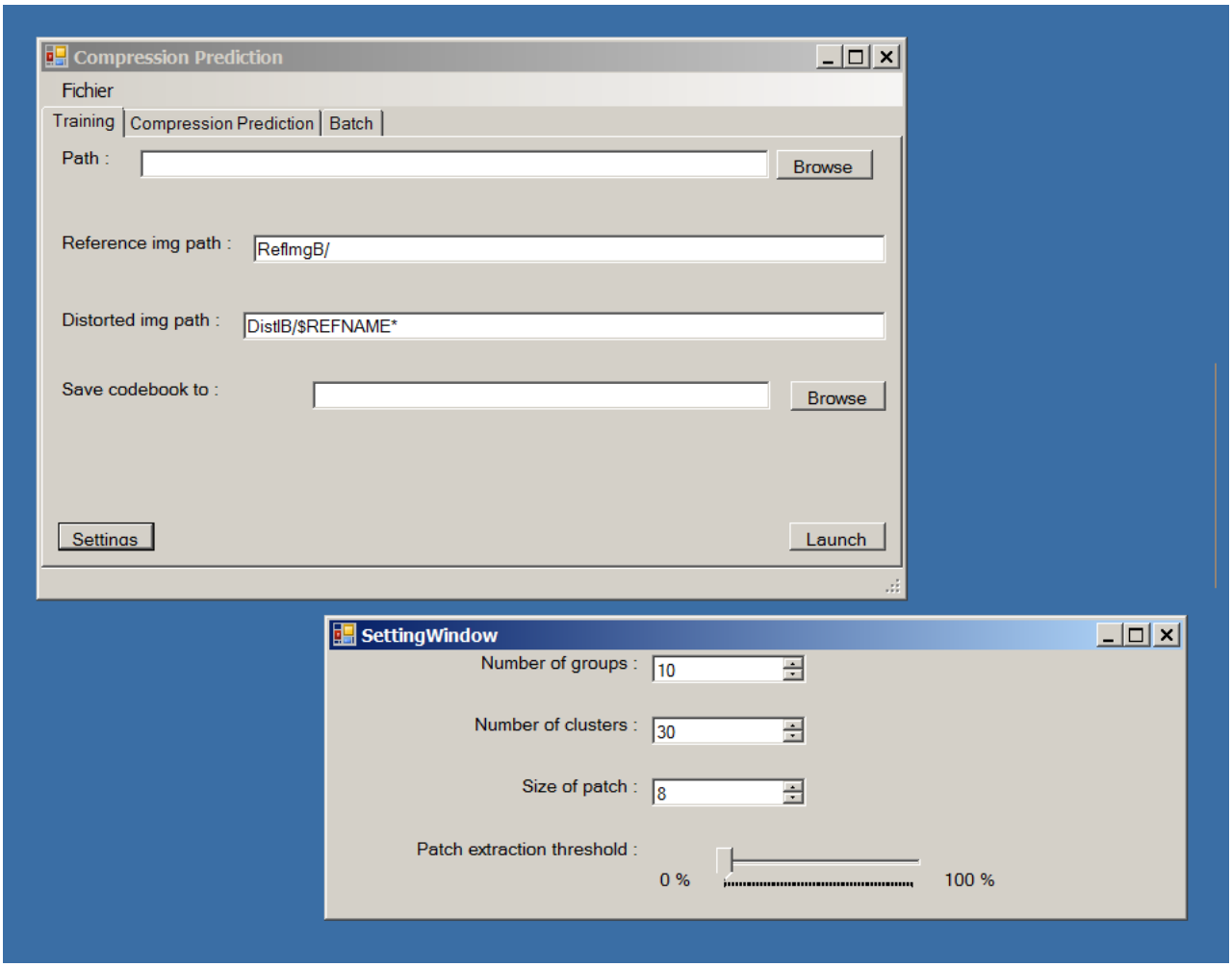


Figure 4.3: GUI : training view

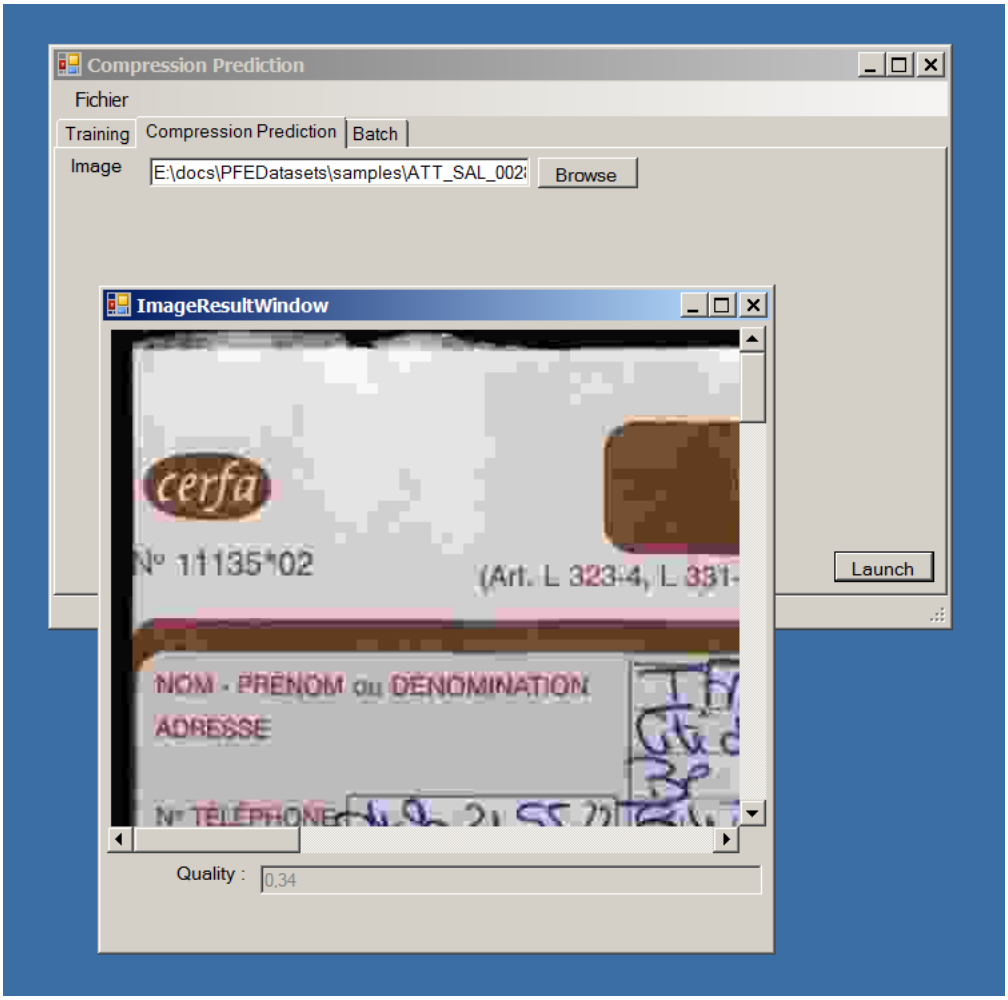


Figure 4.4: GUI : testing view

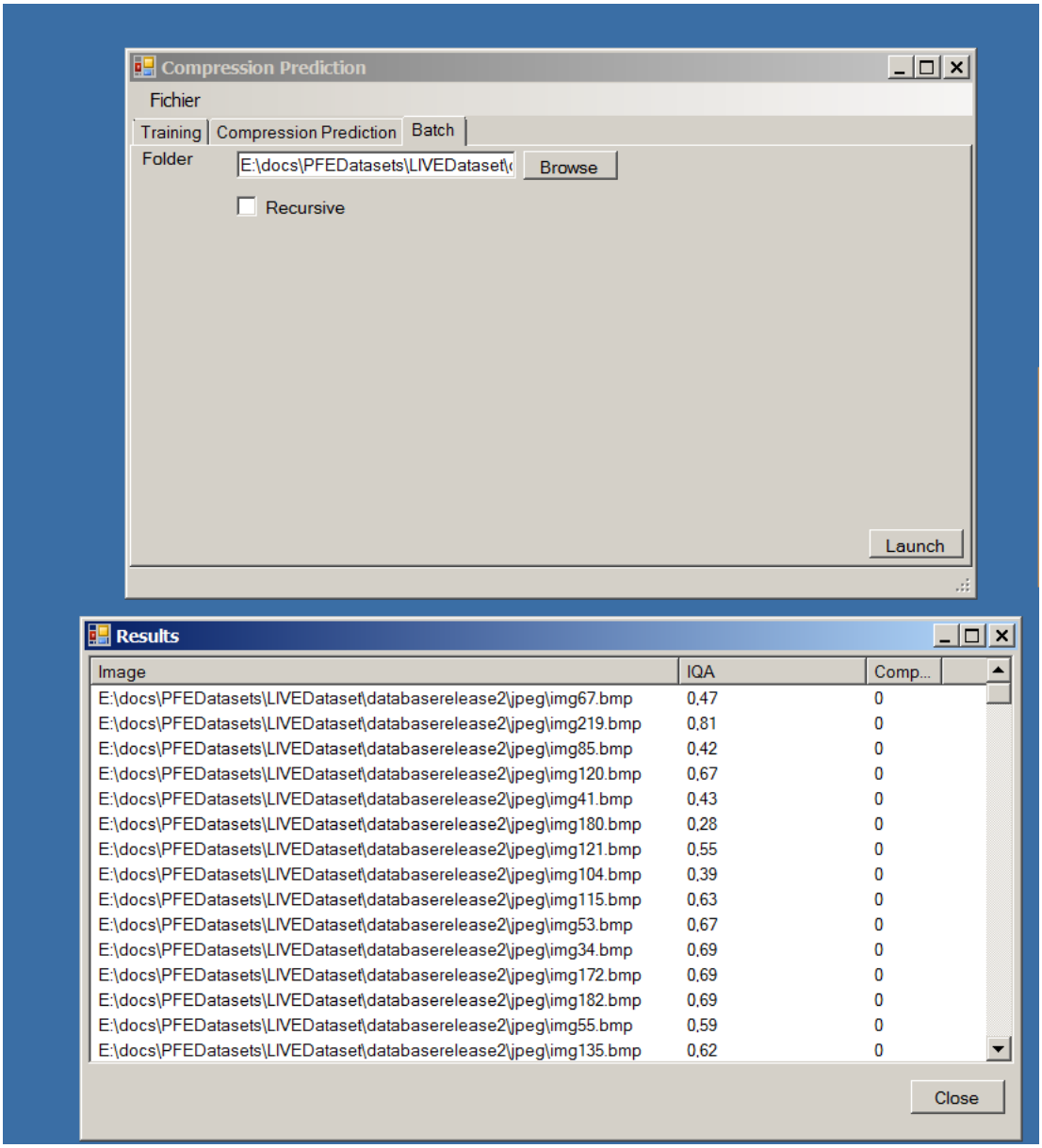


Figure 4.5: GUI : batch view

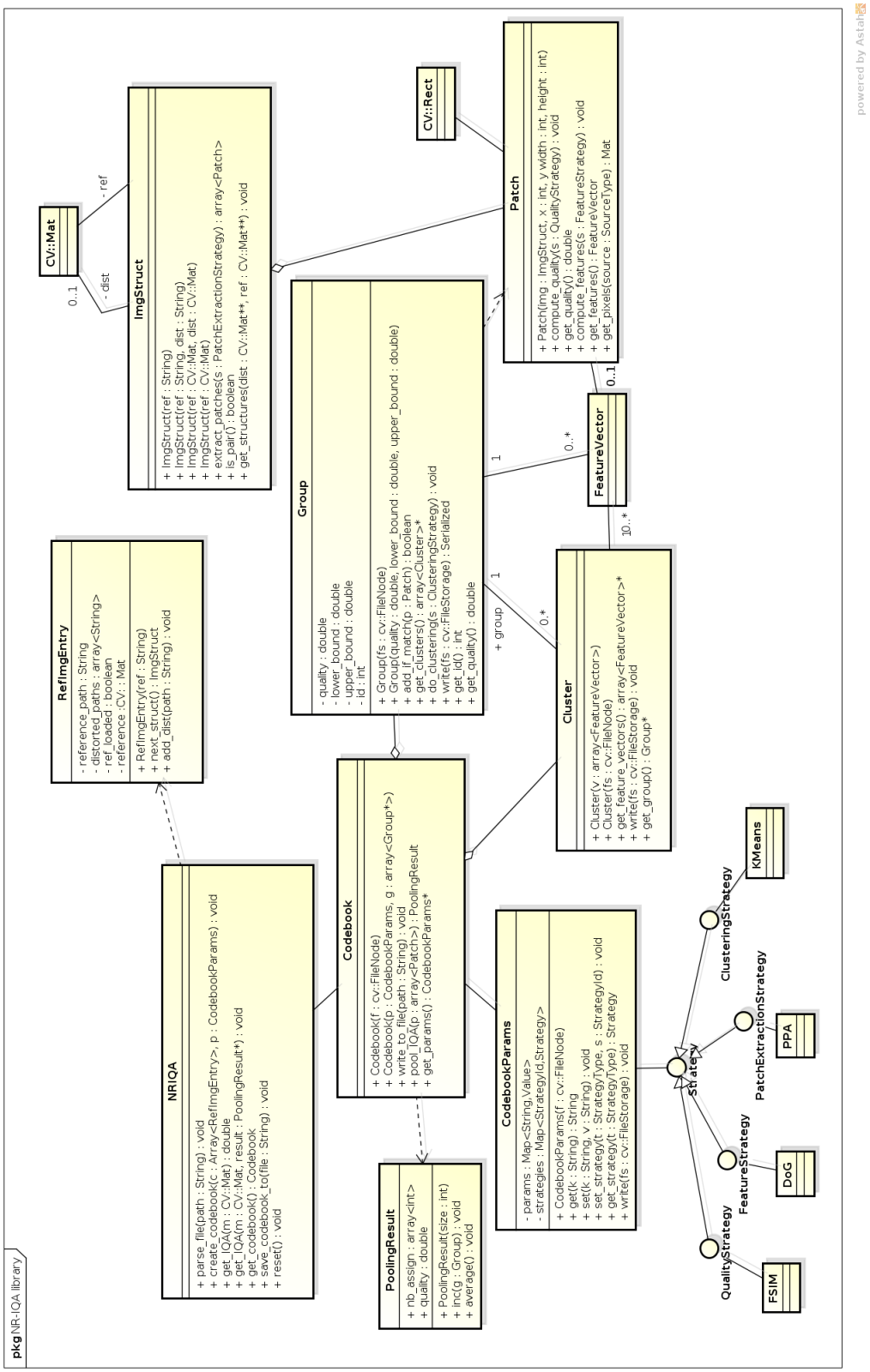


Figure 4.1: The NR-IQA library class diagram

Project management

5.1 Paradigms

5.1.1 V-Model

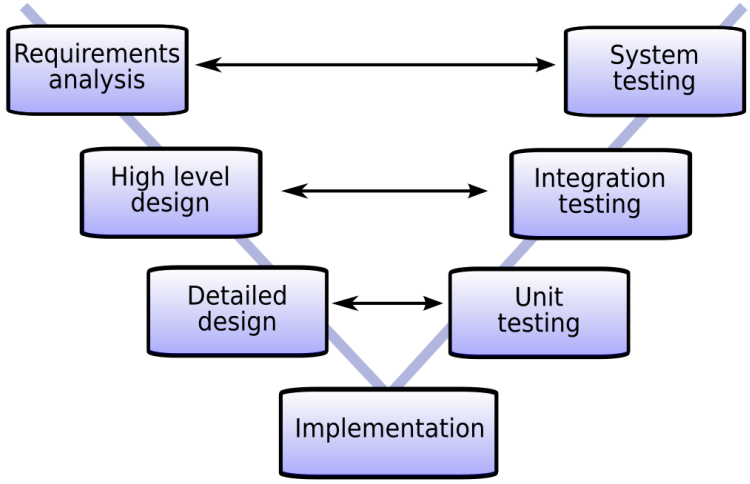


Figure 5.1: The V-Model steps diagram
Source : Wikimedia Commons (http://commons.wikimedia.org/wiki/File:V-model.svg)

The V-Model is a project management method that aims at splitting projects into the following steps.

Requirements analysis Analysis of the client's need

High level design Design of the global system that could fulfill the client's need

Detailed design

Implementation Actually building the system

Unit testing Testing the system's components

Integration testing Ensure that the system's functionalities are operational

System testing Ensure that the resulting system matches the client's need

As shown on diagram 5.1, each design or analysis step is linked to a testing phase. It means that if — during testing — we notice something is wrong, we must go back to the corresponding design step to fix the potential mistakes we did.

5.1.2 UML modeling

During this project, UML modeling tools were used during both specification and design phase.
During the specification, both Use Case and Activity diagrams.
Use Case diagrams are useful to split a system into scenarios or functionalities called *use cases*.
Activity diagrams are relevant when quite complex workflows and algorithms are to be formalized. They can represent parallel or concurrent programs.
During the design phase, UML class diagrams only were used in order to organize and to split responsibilities into classes.

5.1.3 Test Driven Development

Test Driven Development is a workflow that works the following way :

1. Object, function and methods signatures are written
2. Unit-tests are written, taking the aforementioned signatures as black boxes to test
3. Unit-tests can be ran until they pass

This workflow leads to an incentive of wide application testing and prevents code-regression.

5.2 Tools

5.2.1 Trello

Trello is a very minimalistic tool to handle a project on a daily basis. It consists of an enhanced to-do list application which looks a bit like a Kanban board.

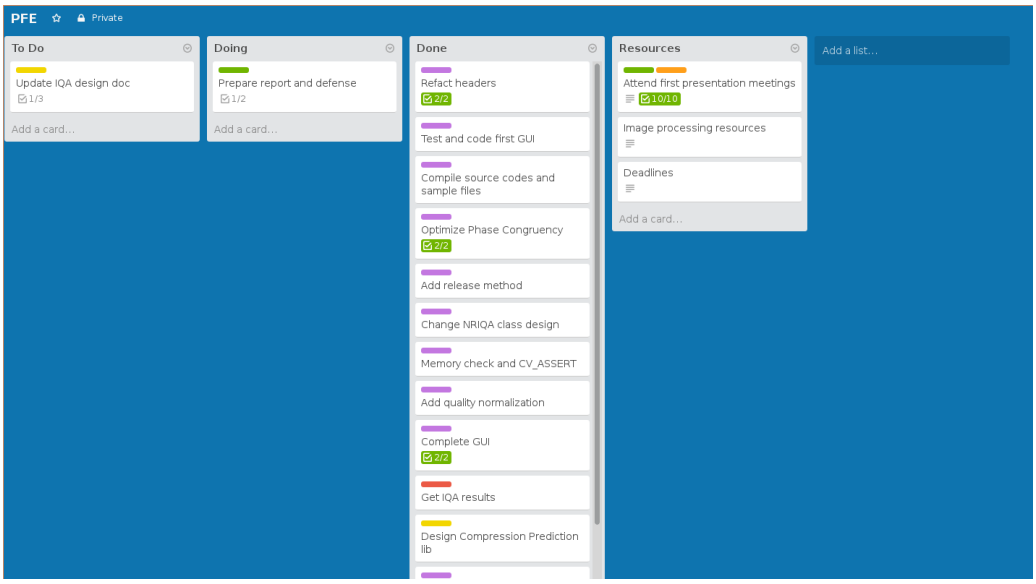


Figure 5.2: The project's trello board (as is at the end of the project)

The grey blocks are lists and the white ones are tasks. You can add as many lists as you need and each task has many attributes to play with such as labels, description or checklists.

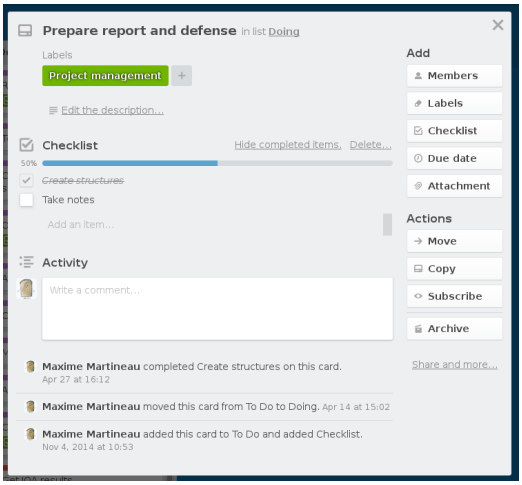


Figure 5.3: A Trello task item (and also a *mise en abyme*)

You can easily move the task items between lists by drag-and-drop which makes the workflow natural, simple and dynamic.

5.2.2 Microsoft Project

Microsoft Project is a project management software. It was mainly used during the project to create Gantt diagrams.

5.2.3 Pencil

Pencil is a mockup and sketching tool. It is useful to easily compose application wireframes.

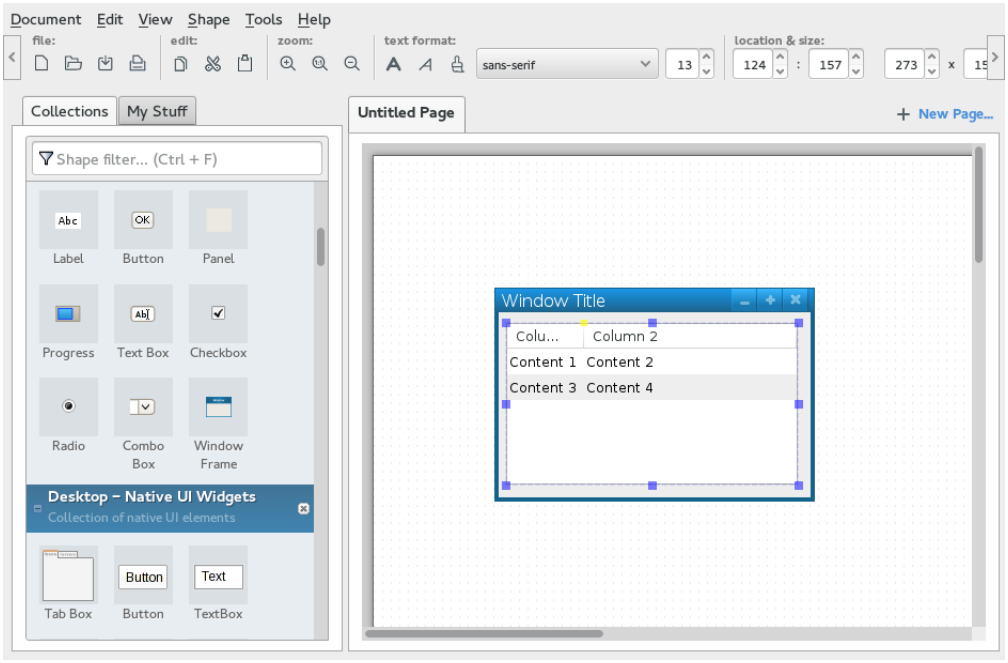


Figure 5.4: Pencil, a mockup tool

5.2.4 Git

Git is a source code management tool as SVN or CVS can be. However, Git offers a decentralized way to handle the code.

5.2.5 Visual Studio Test Explorer

Visual Studio Test Explorer is a tool to follow and manage unit tests suites. Besides other languages, it can handle tests in native C++ and track their status.

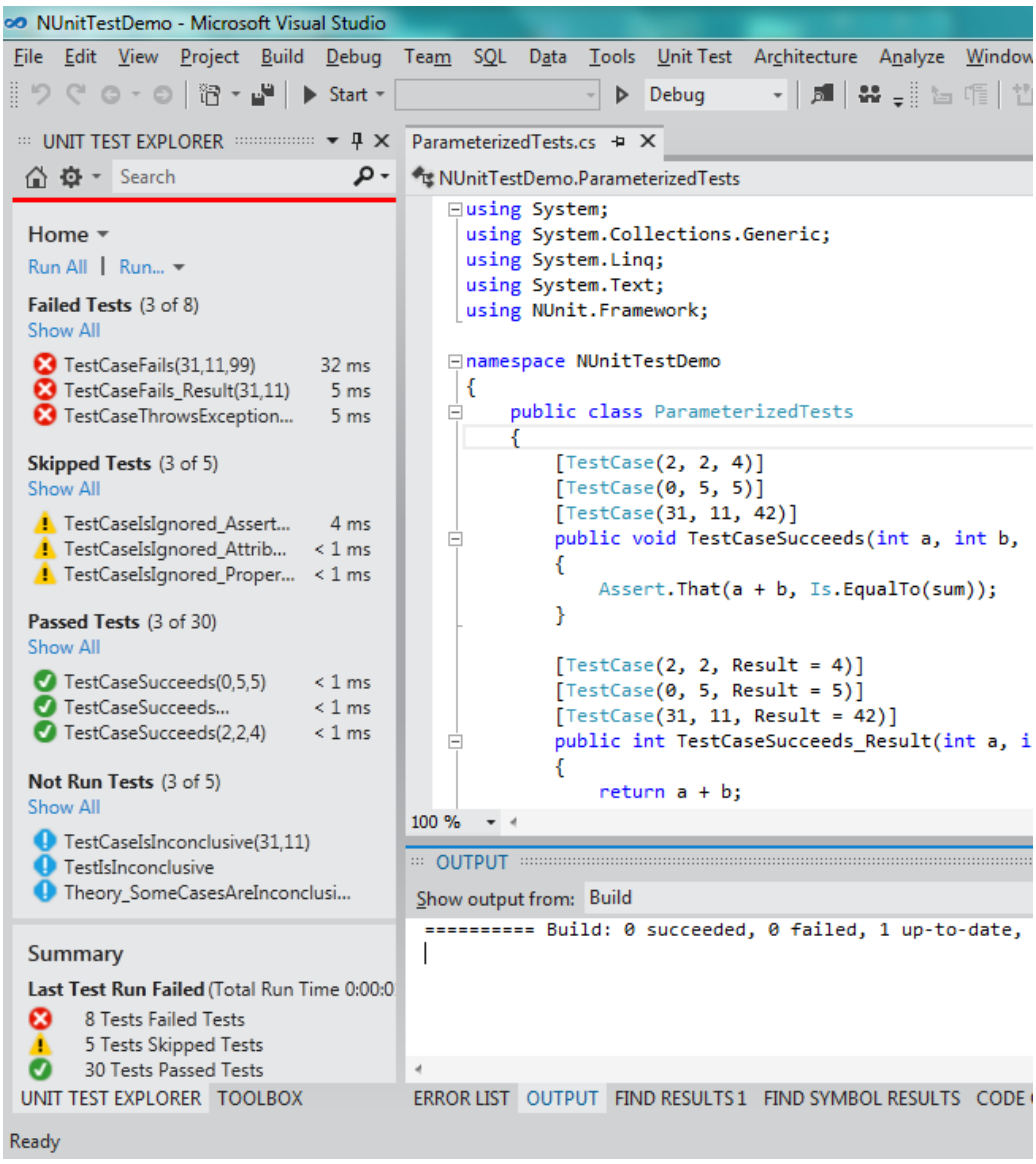


Figure 5.5: Visual Studio Unit Test Explorer

5.2.6 Doxygen

Doxygen is a tool that generates a full-documentation from annotated source code.

5.3 Scheduling

5.3.1 Planned

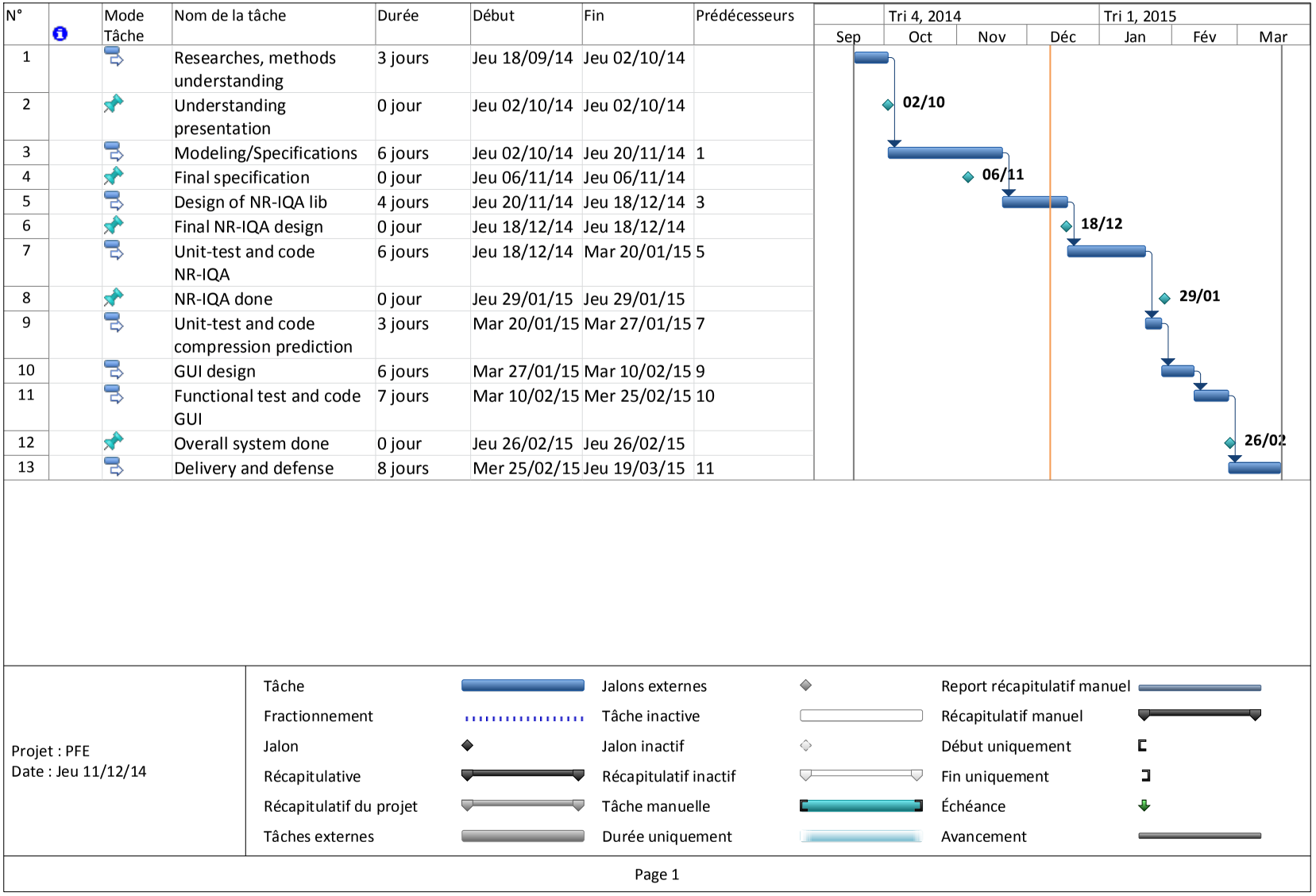
The schedule is planned after the following ideas :

- The overall system must be fully-specified at the beginning
- The design, testing and coding of each pieces of software must be done separately

Therefore, the two libraries and the GUI were planned as three different flows regarding their design, testing and coding.
See figure 5.6

5.3.2 Actual

As I will discuss further, the project sometimes went off deadlines because of some underestimated risks and difficulties.
The main difference between the planned schedule and the actual one is the time on the NR-IQA lib development.
See figure 5.7



N°	Mode Tâche	Nom de la tâche	Durée	Début	Fin	Prédécesseurs
1		Researches, methods understanding	3 jours	Jeu 18/09/14	Jeu 02/10/14	
2		Understanding presentation	0 jour	Jeu 02/10/14	Jeu 02/10/14	
3		Modeling/Specifications	6 jours	Jeu 02/10/14	Jeu 20/11/14	1
4		Final specification	0 jour	Jeu 06/11/14	Jeu 06/11/14	
5		Design of NR-IQA lib	4 jours	Jeu 20/11/14	Jeu 18/12/14	3
6		Final NR-IQA design	0 jour	Jeu 18/12/14	Jeu 18/12/14	
7		Unit-test and code NR-IQA	6 jours	Jeu 18/12/14	Mar 20/01/15	5
8		NR-IQA done	0 jour	Jeu 29/01/15	Jeu 29/01/15	
9		Unit-test and code compression prediction	3 jours	Mar 20/01/15	Mar 27/01/15	7
10		GUI design	6 jours	Mar 27/01/15	Mar 10/02/15	9
11		Functional test and code GUI	7 jours	Mar 10/02/15	Mer 25/02/15	10
12		Overall system done	0 jour	Jeu 26/02/15	Jeu 26/02/15	
13		Delivery and defense	8 jours	Mer 25/02/15	Jeu 19/03/15	11

Figure 5.6: Planned schedule

N°	Mode Tâche	Nom de la tâche	Durée	Début	Fin	Prédécesseurs
1		Researches, methods understanding	3 jours	Jeu 18/09/14	Jeu 02/10/14	
2		Understanding presentation	0 jour	Jeu 02/10/14	Jeu 02/10/14	
3		Modeling/Specifications	6 jours	Jeu 02/10/14	Jeu 20/11/14	1
4		Final specification	0 jour	Jeu 06/11/14	Jeu 06/11/14	
5		Design of NR-IQA lib	4 jours	Jeu 20/11/14	Jeu 18/12/14	3
6		Final NR-IQA design	0 jour	Jeu 18/12/14	Jeu 18/12/14	
7		Unit-test and code NR-IQA	30 jours	Jeu 18/12/14	Jeu 19/03/15	5
8		NR-IQA done	0 jour	Jeu 29/01/15	Jeu 29/01/15	
9		Unit-test and code compression prediction	2 jours	Jeu 19/03/15	Mer 25/03/15	7
10		GUI design	2 jours	Mer 25/03/15	Mar 31/03/15	9
11		Functional test and code GUI	11 jours	Mar 31/03/15	Mer 22/04/15	10
12		Overall system done	0 jour	Jeu 26/02/15	Jeu 26/02/15	
13		Delivery and defense	4 jours	Mer 22/04/15	Jeu 07/05/15	11

Projet : PFE_after
Date : Mar 05/05/15

Tâche
Fractionnement
Jalon
Récapitulative
Récapitulatif du projet
Tâches externes

Jalons externes
Tâche inactive
Jalon inactif
Récapitulatif inactif
Tâche manuelle
Durée uniquement

Report récapitulatif manuel
Récapitulatif manuel
Début uniquement
Fin uniquement
Échéance
Avancement

Page 1

Figure 5.7: Actual schedule

Results and discussion

6.1 Performance and accuracy

6.1.1 Correlation to Ground Truth

The C++ Image Quality Assessment library is wanted to fit the Matlab/Octave implementation in its results (in terms of correlation with the Mean Human Oppinion Score).
Tests were made with the LIVE dataset [7] and the results were similar with :

- Codebook created with Matlab used with C++
- Codebook created with C++ used with C++

This simple test indicates both training and testing phases give similar results.

Nevertheless, the kmeans implementation of OpenCV is not the same as the one in Matlab. There might be differences with other settings or datasets because the clustering centroids are not the same.
That is why another machine learning library is considered : mlpack. Its kmeans implementation supports every options needed to mimic the Matlab one. Moreover, it offers many other machine learning methods.

6.1.2 Compression rate estimation and prediction

There were no scientifical protocol or metric established in order to evaluate the accuracy of the estimation and prediction but it might be good to proceed the following way :
For the compression rate estimation, it might be relevant to compute something like the Mean Square Error between estimated and real compression rates.

For the compression rate prediction, it might be relevant to run Compression Prediction on some set of images, apply the adviced compression and then re-run the Compression Prediction to see whether the compression can be pushed further or not. In an alternative manner, we can plan to have a ground truth for that.

6.2 Project management and handling

This project suffered of a quite important delay. Here might be the reason why the actual schedule did not look like the planned one.

6.2.1 Lack of experience

The fact is that some risks might have been underestimated. The main cause of the delay is the NRIQA library. This library was the most complicated piece of software to develop for its image processing aspect. This project was my first real project in image processing and my first one with Visual C++/OpenCV too. When I had to code the very generic modules, I did not experience any difficulty. Even better it was very enjoyable thanks to the unit testing that enabled me to check my modules one-by-one.

But when it came to code the image processing algorithms, I began to be stuck in quite tedious debugging. That is at this point that I made the deadlines blow up.

6.2.2 Incomplete specifications

When the project began, I felt the needs were clearly defined since it was about implementing a method. After a few months, I was facing inexplicable differences in terms of IQA results. We checked every little piece of the library without finding the bugs.

In fact, I was missing some of the algorithm steps : quality normalization and feature scaling. These details weren't discussed during the meetings but were in the implementations and the given articles.

Doubtlessly, I might have not used this provided element enough, relying too much on what I could have understood from the meetings. One good thing to do could have been to go through these elements more carefully.
One other solution could have been to use an agile method. Since at the beginning of the project needs seemed perfectly clear, there was no doubt that the V-Model was the perfect match for this project.
But what if the things to consider were not only what the clients want but also what you have understood from it ? Maybe Agile could have let me some times to discover every little detail of the algorithms.

6.3 Future work

6.3.1 Further tests on data

Only a few tests were done on datasets. It might be good to perform more tests with more types of datasets and different algorithm settings.

6.3.2 Integration tests

The integration tests — for the GUI — were not clearly defined. A testing document might have been produced.
We can put this issue in perspective, arguing that the GUI won't be used in production but the project management is not complete.

6.3.3 Benchmarking

No tests were made to evaluate the system speed. It might be relevant to come up with tests for different image sizes and patch selection thresholds.

6.3.4 Parallell programming and optimization

The provided version did come with some pieces of code that weren't using the OpenCV multithreading abilities.
An alternative version will be provided.

Conclusion

This project introduced to new languages and tools such as OpenCV, Visual C++ and C++/CLI.
It allowed me to have a first experience in image processing programming and it introduced me to this research area.
I also had the opportunity to present our work at a research workshop organized by Itesoft. During this workshop, I also had the opportunity to hear about some related work in the field of document image analysis and document dematerialization.
Finally, a complete system was produced, fully-working and fully-tested.

Bibliography

[1] A. Alaei, D. Conte, and R. Raveaux. Document image quality assessment based on improved gradient magnitude similarity deviation.

[2] A. Alaei, D. Conte, and R. Raveaux. Image quality assessment with a specific focus on document images.

[3] A. Alaei, P. Nagabhushan, and U. Pal. Piece-wise painting technique for line segmentation of unconstrained handwritten text: a specific study with persian text documents. *Pattern Analysis and Applications*, 14(4):381–394, 2011.

[4] E. C. Larson and D. M. Chandler. Most apparent distortion: full-reference image quality assessment and the role of strategy. *Journal of Electronic Imaging*, 19(1):011006, 2010.

[5] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[6] N. Ponomarenko and K. Egiazarian. Tampere image database 2008 tid2008, 2009.

[7] H. Sheikh, Z. Wang, L. Cormack, and A. Bovik. Live image quality assessment database release 2 (2005).

[8] W. Xue, L. Zhang, and X. Mou. Learning without human scores for blind image quality assessment. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 995–1002. IEEE, 2013.

[9] W. Xue, L. Zhang, X. Mou, and A. Bovik. Gradient magnitude similarity deviation: A highly efficient perceptual image quality index. 2014.

[10] L. Zhang, D. Zhang, and X. Mou. Fsim: a feature similarity index for image quality assessment. *Image Processing, IEEE Transactions on*, 20(8):2378–2386, 2011.

Image Quality Assessment : C++ Implementation and testing

Département Informatique
5ième année

Final year project report

Résumé : Ce projet de fin d'étude vise à implémenter un système d'estimation de la qualité et de la compression des images. L'algorithme à implémenter est dit "sans référence" et peut apprendre tout type de distortion visuelles.

Mots clefs : image, qualité, estimation, Visual C++, OpenCV, Windows Forms

Abstract: This final year project aims at implementing an Image Quality Assessment and compression level estimation system. The algorithm to implement is "no-reference" and can learn to recognize every type of visual distortions.

Keywords: IQA, image, quality, estimation, compression estimation, Visual C++, OpenCV, Windows Forms

---

Supervisors
Alireza Alaei
alireza.alaei@univ-tours.fr
Donatello Conte
donatello.conte@univ-tours.fr
Romain Raveaux
romain.raveaux@univ-tours.fr

Student
Maxime Martineau
maxime.martineau@etu.univ-tours.fr

DI5 2014 - 2015

Université François-Rabelais, Tours