DEPARTMENT OF
MECHANICAL AND INDUSTRIAL
ENGINEERING

MIGUEL ÂNGELO OLIVEIRA BATISTA
⟨Author Degree UNDEFINED⟩

# DEVELOPMENT AND APPLICATION OF THE DIGITAL IMAGE CORRELATION TECHNIQUE IN PYTHON CODE

## MONITORIZATION AND CHARACTERIZATION OF MATERIALS AND STRUCTURES

MASTER IN ⟨MSC PROGRAM NAME⟩

NOVA University Lisbon
⟨month⟩, ⟨year⟩

DEPARTMENT OF
MECHANICAL AND INDUSTRIAL ENGINEER-
ING

# DEVELOPMENT AND APPLICATION OF THE DIGITAL IMAGE CORRELATION TECHNIQUE IN PYTHON CODE

## MONITORIZATION AND CHARACTERIZATION OF MATERIALS AND STRUCTURES

**MIGUEL ÂNGELO OLIVEIRA BATISTA**

⟨Author Degree UNDEFINED⟩

**Adviser:** Mary Doe Adviser Name
*Full Professor, NOVA University Lisbon*

**Co-advisers:** John Doe Co-Adviser Name
*Associate Professor, NOVA University Lisbon*
John Doe other Co-Adviser Name
*Full Professor, NOVA University Lisbon*

MASTER IN ⟨MSC PROGRAM NAME⟩

NOVA University Lisbon
⟨month⟩, ⟨year⟩

**Development and application   of the digital image correlation technique in python code**

*Dedicatory lorem ipsum.*

# Acknowledgements

Acknowledgments are personal text and should be a free expression of the author.

However, without any intention of conditioning the form or content of this text, I would like to add that it usually starts with academic thanks (instructors, etc.); then institutional thanks (Research Center, Department, Faculty, University, FCT / MEC scholarships, etc.) and, finally, the personal ones (friends, family, etc.).

But I insist that there are no fixed rules for this text, and it must, above all, express what the author feels.

*"You cannot teach a man anything; you can only help him discover it in himself." (Galileo)*

# Abstract

Regardless of the language in which the dissertation is written, a summary is required in the same language as the main text and another summary in another language. It is assumed that the two languages in question are Portuguese and English.

The abstracts should appear first in the language of the main text and then in the other language. For example, if the dissertation is written in Portuguese the abstract in Portuguese will appear first, then the abstract in English, followed by the main text in Portuguese. If the dissertation is written in English, the abstract in English will appear first, then the abstract in Portuguese, followed by the main text in English.

In the LaTeX version, the NOVAthesis template will automatically order the two abstracts taking into account the language of the main text. You may change this behaviour by adding

```
\abstractorder(<MAIN_LANG>):={<LANG_1>,...,<LANG_N>}
```

to the customization area in the document preamble, e.g.,

```
\abstractorder(de):={de,en,it}
```

The abstracts should not exceed one page and, in a generic way, should answer the following questions (it is essential to adapt to the usual practices of your scientific area):

1. What is the problem?
2. Why is this problem interesting/challenging?
3. What is the proposed approach/solution?
4. What results (implications/consequences) from the solution?

**Keywords:** Keyword 1, Keyword 2, Keyword 3, Keyword 4, Keyword 5, Keyword 6, Keyword 7, Keyword 8, Keyword 9, ...

# Resumo

Independentemente da língua em que a dissertação esteja redigida, é necessário um resumo na mesma língua do texto principal e outro resumo noutra língua. Pressupõe-se que as duas línguas em questão sejam o português e o inglês.

Os resumos devem aparecer primeiro na língua do texto principal e depois na outra língua. Por exemplo, se a dissertação for redigida em português, o resumo em português aparecerá primeiro, seguido do resumo em inglês (*abstract*), seguido do texto principal em português. Se a dissertação for redigida em inglês, o resumo em inglês (*abstract* aparecerá primeiro, seguido do resumo em português, seguido do texto principal em inglês.

Na versão LaTeX o template NOVAthesis irá ordenar automaticamente os dois resumos tendo em consideração a língua do texto principal. É possível alterar este comportamento adicionando

```
\abstractorder(<MAIN_LANG>):={<LANG_1>,...,<LANG_N>}
```

à zona de customização no preâmbulo do documento, e.g.,

```
\abstractorder(de):={de,en,it}
```

Os resumos não devem ultrapassar uma página e, de forma genérica, devem responder às seguintes questões (é essencial adaptá-los às práticas habituais da sua área científica):

1. Qual é o problema?
2. Porque é que é um problema interessante/desafiante?
3. Qual é a proposta de abordagem/solução?
4. Quais são as consequências/resultados da solução proposta?

**Palavras-chave:** Palavra-chave 1, Palavra-chave 2, Palavra-chave 3, . . .

# Contents

# List of Figures

# List of Tables

# Glossary

# Acronyms

# Symbols

# 1 Glossary

On the course of this paper, there will be many words/expressions that are mainly used in therms of programming. In order to help the reader, here are the most common words that will appear in the paper and its definitions.

## 1.1 IDE

An integrated development environment (IDE) is a software application that provides the necessary means for software development. It normally consists of a source code editor, build automation tools, and a debugger. For the development of the NAME OF SOFTWARE CREATED, the used IDE was the Spyder (as well as for the development of the GUI//and for the development of the GUI it was used the XXXX).

## 1.2 GUI

A Graphical User Interface (GUI) is the first thing the user sees and interacts with when opening an application or website. It allows the user to interact with the application through the help of buttons, graphics, and such instead of the need for coding. Depending on how it is designed, it might enhance the users experience with the application or worsen it, therefore/ence its importance.

## 1.3 Library

A Library is, in a simple manner of speech, a collection of related modules. Modules, on the other hand, are files containing specific python functions, definitions, and statements that can be called into/upon (?) the main program without the need of rewriting its code, making it easier and more convenient for programmers to code. Some of the most known libraries that will be discussed during this paper are, i.e., Tkinter (Standard GUI library for Python); MatPlotLib (Useful for its visual characteristics).

## 1.4 Module

## 1.5 Framework

A framework is a

## 1.6 Cross-Platform Framework

A cross-platform app development framework is a set of tools that allows you to use a single codebase to build native or native-like apps for multiple platforms such as Android, iOS, Desktop and Web.

## 1.7 Open Source

Open Source, originaly Open Source Software (OSS), is the therm used for codes that are meant to be publicly accessible. In other words, open source codes are codes which anyone can see, modify and distribute as they please.

## 1.8 API

API stands for Application Programming Interface, which is a set of definitions and protocols for building and integrating application software

2

# Software Choice

## 2.1 Criteria of choice

In order to choose the perfect software and library to build the GUI, the following criteria, which will then be summarised, were taken into account:

- License and copyright issues;
- Cross-platform compatibility;
- Libraries compatibility;

### 2.1.1 License and Copyright issues

When the program is finished, in order to be able to use the software as it is pleased, it is necessary that the codes and libraries, used in aid of building the GUI, are all open-source and of free use to its users. For that purpose, there are licenses that may restring certain rights to the developer or give full copyright (of it)(?) to the owner. Having that in mind, the software licenses that will actually be of use to the project can be divided into the following categories:

- **Public Domain**
  This type of license is the most permissive, granting all rights to the user, thus RELINQUISHING/RENOUNCING(?) any kind of copyright.
  In other words, this type of license allows the user to adopt the code and reuse the software as he pleases.

- **Permissive**
  Permissive licenses are the most common among open-source software licenses. It carries only minimal restrictions on how the software can be used, modified, and redistributed, generally requiring only that the original copyright notice be retained.
  **Examples of Permissive Licenses -** MIT License, BSD licenses, Apache license.

- **Copyleft**
  A copyleft license is very similar to a permissive license with the main difference being that it doesn't give the user the right to sublicense.
  For example, if the original project was open-source and was under a copyleft license, any modified versions of it should also be open-source and be under the same copyleft license. In contrast, if the original project was under a permissive license, the user may sublicense it under the type of license he desires.
  **Examples of Copyleft Licenses -** General Public License (GNU/GPL), Linux kernel, OBS.

- **Lesser General Public License [LGPL]**
  The LGPL is not really a category of licenses, but a license itself, and is (SOMEWHAT/KINDA/-?) the middle term between permissive and copyleft licenses.(ALTERAR/RESCREVER?)
  If a developer uses a LGPL-licensed library to aid his code, he is allowed to license his project under any kind of license of his desire, just like a permissive license. However, if the developer modifies such library or copy parts of it into is code, he will be obligated to put his project under the same LGPL license, similar to copyleft licenses.

- **Proprietary**
  Proprietary license or proprietary software is the most restrictive kind of license. It gives MOST (?????) of the rights to the developer, making the software ineligible for copying, modifying, or distribution.

For the development of this project, it was intended that the developed software would be open-source and, therefore, have no copyright restrictions. As such, the type of license of the chosen software used in the building of the GUI should either be a public domain, permissive, copyleft, or LGPL license.

### 2.1.2 Cross-platform Compatibility

Being the software a tool intended for research and professional means, the possibility of a user not being able to use it, because it is not compatible with his current operating system, is undesired. AS SUCH, the chosen library should be able to CREATE a cross-platform GUI, compatible with the BIGGEST computer operating systems, namely Windows, Linux, and MacOS.

### 2.1.3 Libraries Compatibility

Since the main purpose of the software is to apply techniques of digital correlation for the monitoring and characterization of materials and structures, it is crucial, for its development, that the chosen module offers a good range of functionalities in terms of image manipulation.

As such, the module needs to be capable of READING images under the format .tif, which stands for **T**agged **I**mage **F**ile, and be compatible with the following libraries:

- **OpenCV -** Responsible for calling the .tif files;
- **MatPlotLib -** Responsible for turning the images into matrix;
- **NumPy** - Responsible for the ANALYSIS? ;

which are responsible for the graphical analysis of the uploaded files.

One of the difficulties that Tkinter, the standard Python module for GUI construction, presented when building the first prototype, was shown while executing the analysis, where it would open a new pop-up window for each of the uploaded image files ,overloading the users screen with pop-ups, instead of only showing the intended result embedded on the main window of the GUI.

## 2.2 Choosing the library

In order to choose the appropriate library, it was taken into account the opinions and questions of many developers that were shared in websites like Stackoverflow and Quora, websites whose main purpose is to SHARE question and answers from the community. Since coding is not really the main focus of a masters in mechanical engineering, it was REASSURING THAT the library used in the development of the GUI would have a good BASE COMMUNITY that might have already had the same problems that could occur in the process.

A good example of one of the libraries that really seemed promising but "DINDNT MAKE THE FINAL CUT", because of its community, was **Dear PyGui**, based on Dear ImGui a well known GUI library for C++. Although looking really appellative in terms of graphical components AND SUCH(?), it is very recent in comparison with many other good libraries, with its first version (0.1.3) being released in April 2020, and, because of that, it lacks a good BASE COMMUNITY to rely on in case of needing help, which could turn out to be problematic.

Other libraries that were considered but didn't make the final cut, either for technicalities or sheer preference, were:

- **Kivy** - Kivy offers modern graphics and design techniques. It is most known for its versatility in cross-platform compatibility, being able to run in Android, iOS, Linux, macOS, and Windows. Since the developed software is not meant to work on Android or iOS, at least not in THIS/A early stage, it was excluded from the list.

(ESTÁ BOM ASSIM?)

- **wxPython** - Acting as wrapper for the wxWidgets framework, it allows the developer to create native user interfaces (NUI, interfaces similar to the OS they are used in. Although it could be easier to go for a native UI, it was MY PERSONAL CHOICE/PREFERENCE to be able to have full control of the aspect that the GUI would have regardless of its operating system.

- **Tkinter** - Tkinter, as it was mentioned before, is Python's standard GUI and was the one used to build the software prototype. Since, in the building of the prototype, there were some undesired INCONVENIENCES, that would turn the code HEAVIER than the necessary, it was excluded from the list in order to find a better solution.

- **PySimpleGUI** - Just like the name implies, PySimpleGUI is a module designed to ease up a little bit the process of creation of a GUI. It is based on some of the already mentioned frameworks, namely wxPython, Qt, and Tkinter, and is ideal for NEWBIES/STARTERS. Fearing that, in order to make it easier for the developer, might mean that some functionalities of the based framework were cut out, it was automatically a hard no for this one. (ALTERAR/OUTRAS PALAVRAS)

- **Wax** - Similar to PySimpleGUI, Wax is a wrapper for wxPython and was excluded from the list for the same mentioned reason.

Given the big list of libraries available, the final deliberation ended up being between **PyQt6** and **PySide6**. Both these libraries have Qt as base framework and are (said to be) very similar in terms of coding, being the major deliberating (???) factor for many developers the difference in their licenses, where PyQt6 has a GPL or commercial license and PySide6 has a LGPL license.
One of the major factors in favour of both MODULES was their compatibility with QtDesigner. QtDesigner is a tool, also from the Qt Company, that allows the creation of the GUI with a drag-and-drop system, which allows the developer to customise its GUI in a what-you-see-is-what-you-get (WYSIWYG) manner, without the need of always resorting to the MAIN code.
Being both very similar, in the end what drew more attention was the fact that, at the current date, PySide6 is the official Qt library for Python, which should ensure its viability in a long term.

# 3 Digital Image Correlation

## 3.1 What is it?

Digital image correlation, DIC, is an optical full-field strain measurement technique for 2D or 3D ANALYSIS. IN A SIMPLER MANNER OF SPEECH, DIC consists in the comparison of two images of an object before and after its deformation.
This technique has first been applied to digital images in 1975 and, thanks to the progress of technology and improvement of cameras, it has become a reliable technique for many present-day applications, like image analysis, velocimetry, and strain estimation.

Normally, in order to perform the technique, the sample surface is painted with a solid base colour, normally black or white, and then SPRINKLED/SPRAYED WITH A CONTRASTING COLOUR, creating distinct patterns all over it. Those patterns are then taken as reference in the control, non-deformed, picture, and compared, in terms of displacement and DEFORMATION, with the patterns of the following deformed pictures, from which the strains can also be calculated through.

This method of analysis is possible by transforming the pictures into matrices, where each of its units represent subsets or blocks of pixels, correlating the position of such pixels in the original and deformed pictures, as shown in the figure 3.1, hence the need of creating distinct patterns. It also allows the

The developed software is a 2D DIC and

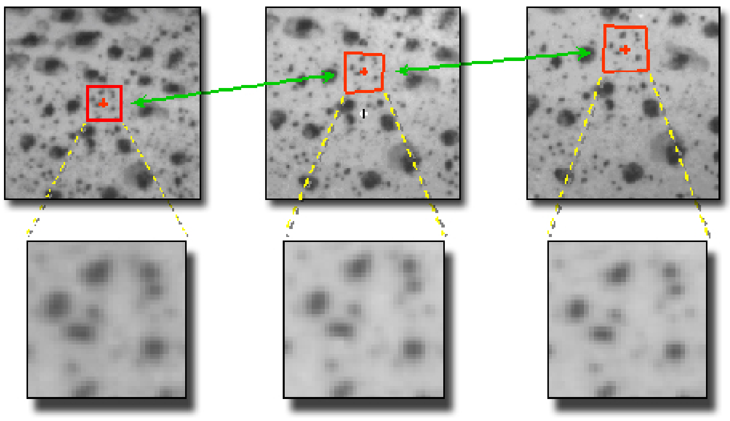
Figure 3.1: Example of image correlation