RTCorr

Real-Time Correlation Software

Instruction Manual

8/2018

Jakub Antoš and Václav Nežerka

Department of Mechanics, FCE, CTU in Prague

jakub.antos@fsv.cvut.cz

Introduction

The software tool RTCorr (Real-Time Correlation) for real-time optical measurement of displacements was developed in Python 3.5. It offers a graphical user interface (GUI) for placement and management of virtual extensometers, setting DIC parameters, and export of data. The program is capable of accomplishing simultaneous measurements of relative displacements in 2D using an arbitrary number of virtual extensometers, equivalent to physical ones. A full-field 2D DIC analysis can be carried out once the measurement is finished from the acquired images that are saved (if required) to a project folder. The measurement can be scaled to physical units. Otherwise, it is carried out in pixels.
The source files and manual with a thorough description of the program structure and GUI controls are provided at the author's GitHub repository website.

Sub-pixel registration
The software employs the upsampled matrix-multiplication discrete Fourier transform (DFT), proposed by Guizar-Sicairos et al. [1], to reach subpixel accuracy. Using this approach, cross correlation for each square subset of pixels of a side length N is accomplished in a frequency domain [2, 3], rendering the calculation robust and relatively insensitive to noise. Since the pixel subsets are registered by phase retrieval, only their translation is evaluated, and the displacement field is averaged over the subset area.
The algorithm is capable of computing subset shifts within a fraction of a pixel, 1/κ.

1. Program Structure

The program initiates after running the main.py script, located in the root program folder (Figure 1). Exported data (images, text files) can be found in the "output". Folder "graphic" and "__pycache__" store auxiliary files and do not need to be accessed by the user.

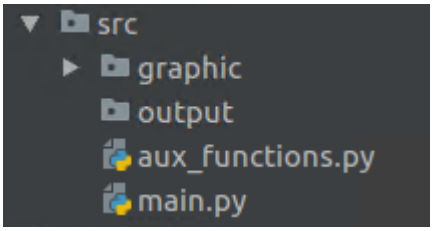

Figure 1: Structure of the program folders and files.

The program can be run via *main.py* from a command prompt (Windows) / terminal (Mac, Linux) by calling python3, see Figure 2. In order to run the program successfully, a few packages (besides Python 3.5+ must be installed. These include **os, time, tkinter, PIL, matplotlib, copy, subprocess, numpy, pypylon[1], image registration[2]** and **sys**. The information on how to install packages on your computer can be found, e.g., here.

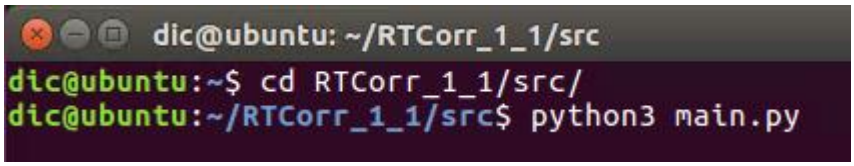

Figure 2: Navigation into the program folder and running main.py.

2. Program workflow

The program workflow is very intuitive and the user can adjust all parameters using GUI controls (see Figure 3). After running the main.py the main GUI window appears.

All the parameters, such as extensometer positions, subset size, sampling rate, etc., can be adjusted using edit boxes (entry widgets) – however, after entering a value **THE ENTER KEY MUST BE PRESSED,** otherwise the entered value has no effect. The user can check the correct input value in the "Messages" log box.

Initialize camera

First step is camera initialization. Program is currently able to communicate with with any Basler USB 3.0 *ace* family camera. Automatic camera detection works in case of one connected camera only. After a successful initialization, a current picture appears in the left part of the screen window.

Figure 3: PyRTC GUI default state.

Reload camera image

Manually reloads the camera image. Useful tool during the precise focusing onto the testing object surface contrast pattern.

Get scale

Set to the physical dimension function. By picking two individual points and defining real scale to get px/mm ratio.

Place extensometers

Opens a window with an option to place a pair of individual extensometers. Extensometers can be individually placed anywhere within the image region and coordinates of each point can be edited via the edit boxes located at the right-bottom.

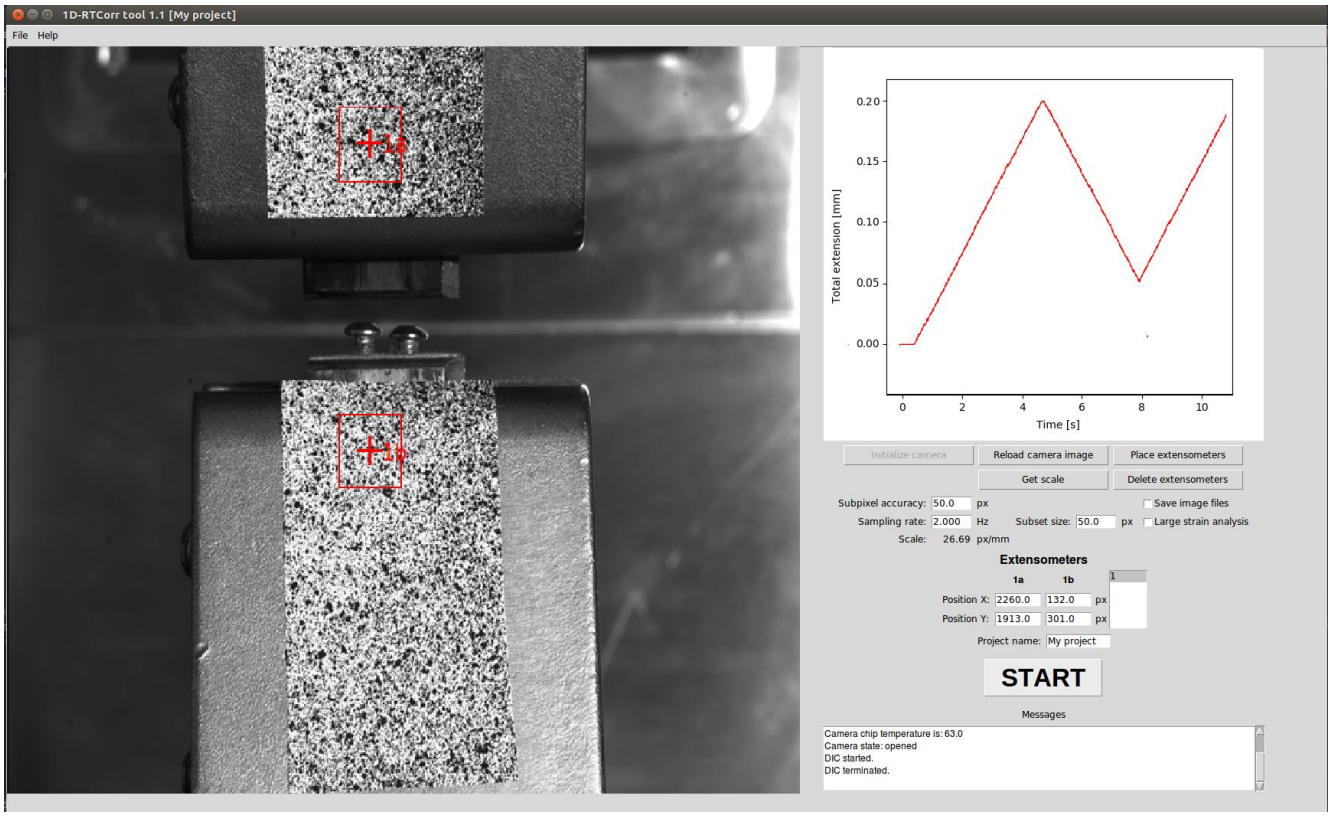

Figure 4: PyRTC GUI during the experimental testing.

Delete extensometers

Deletes the selected set of extensometers.

Subpixel accuracy [px]

Defines the finer accuracy than that of the pixel width, at a fraction of the pixel size.

Sampling rate [Hz]

Sets the number of measurement cycles per second.

Start

Starts the main loop.

□ ***Save image files***

Optional saving of acquired pictures into the \Output\MyProject_folder.

□ ***Large strain analysis***

Correlation in between two following pictures. Previous image is used as a reference. Without checking this box (small strain analysis), the first image of a sequence is used as the reference one.

3. Program loop

After starting stopping the main loop, program will create an output text file in \output\My project-YYYY_MM_DD-HH_MM_SS_results_My project.txt. Output file contains sets of data in following format:

time	lag	dx_1	dy_1	disp_P1(x)_1	disp_P1(y)_1	disp_P2(x)_1	disp_P2(y)_1
[s]	[s]	[mm]	[mm]	[mm]	[mm]	[mm]	[mm]

Where:

Time – time from launching the correlation process and calculation of displacements

Lag – delay in between the timing at the set frequency and true time of acquiring individual images (if the frequency is higher than the hardware capabilities)

dx_1 – extensions of extensometers in horizontal direction

dx_1 – extensions of extensometers in vertical direction

disp_P1(x)_1 – displacement of extensometers, first point, horizontal direction

disp_P1(y)_1 – displacement of extensometers, first point, vertical direction

disp_P2(x)_1 – displacement of extensometers, second point, horizontal direction

disp_P2(y)_1 – displacement of extensometers, second point, vertical direction

Acknowledgement

The support by the internal grant of the Faculty of Civil Engineering at CTU in Prague [grant 154 number SGS18/037/OHK1/1T/11] is gratefully acknowledged.

References

[1] M. Guizar-Sicairos, S. T. Thurman, J. R. Fienup, Efficient subpixel image registration algorithms, Optics Letters 33 (2008) 156–158. doi:10.1364/ol.33.000156.

[2] J. R. Fienup, Reconstruction of an object from the modulus of its Fourier transform, Optics Letters 3 (1978) 27–29. doi:10.1364/ol.3.000027.

[3] J. R. Fienup, Phase retrieval algorithms: a comparison, Applied Optics 21 (1982) 2758–2769. doi:10.1364/ao.21.002758.