An open-source digital image correlation acquisition system, focusing on temporal synchronization with the Gleeble 3800

Maxwell Vos[1*], *David* Dallas[1], *James Anthony* Dicks[1], *Thorsten Hermann* Becker[1], and *Sarah Louise* George[1]
[1]Centre for Materials Engineering, Department of Mechanical Engineering, University of Cape Town, South Africa

Abstract. Current commercial digital image correlation (DIC) systems offer limited versatility, particularly with regards to integrating DIC and thermal imaging on the Gleeble 3800, and are also often financially prohibitive. On the other hand, current open-source acquisition solutions lack the specifications required for most DIC applications and custom solutions are complex for most DIC users. This research therefore aimed to develop an open-source DIC acquisition system for materials research applications that fulfil these shortcomings. The DIC Capture system addresses these limitations by using high quality hardware that interfaces with user orientated software, thus facilitating complex material testing protocols at a reduced cost.

1 Introduction

The Gleeble 3800 is a thermomechanical testing apparatus capable of forces up to 200 kN at stroke rates of 2000 mm/s, which can be performed under vacuum at temperatures up to 3000 °C with heating and cooling rates of up to 10000 °C/s by using either ohmic or induction heating [1]. For this reason, the Gleeble 3800 is a suitable instrument to simulate industrial processes (e.g., hot rolling , welding [2]), creep [3, 4] and fatigue scenarios [5], as well as for the evaluation of high-performance applications [6, 7]. Although these capabilities are useful to ascertain bulk mechanical properties at elevated temperatures, it is often useful to gain information about localised strain phenomena. One such method of achieving such is using digital image correlation (DIC). DIC is a non-contact optical measurement method for quantifying the displacement and deformation of multiple points on a sample's surface, and has become a widely accepted and commonly used technique for the quantification of localised strain behaviour in mechanical testing [8]. Compared to alternative strain measurement techniques such as strain gauges, optical fibre sensors, and clip gauges, DIC has proved to be an effective strain measurement technique for high temperature testing, primarily due to its robust and contactless attributes [9]. In particular, 3 dimensional (3D) (full-field) DIC also allows for the measurement of out-of-plane effects owing to its use of two (or more) cameras [10]. Although there currently exist multiple open-source DIC

* Corresponding author: maxwellvosthefirst@gmail.com

software tools for the analysis of DIC data [11], the development of systems to effectively generate the raw data needed for these DIC analysis tools has received little attention. In particular, developing a system that facilitates through-test variable acquisition rates is useful for data management. Additionally, when performing high temperature DIC without simultaneous thermal imaging, a uniform or assumed temperature profile has to be used to calculate temperature dependant material properties [12]. To overcome this issue, temporally synchronised thermal mapping can be used in conjunction with DIC, thus allowing for measurement of both localised mechanical behaviour and accurate temperature profiles across the specimen.

Given the benefits that synchronised full-field DIC and thermal acquisition may present to understand advanced mechanical behaviour at elevated temperatures, an open-source system capable of through-test variable acquisition rates was developed and integrated with the Gleeble 3800 system and using MatchID software [13] for analysis.

2 Methodological approach

2.1 Hardware setup

The DIC Capture system comprises of a hardware data acquisition (DAQ) board (Fig. 1a), full-field camera system, thermal camera and control computer (Fig. 1b). The ADC ADS8584S from Texas Instruments was chosen for this application (see Appendix A1 for specifications). The DAQ was controlled by an Arduino Nano ESP32 that sends camera trigger times, voltage values from the ±10V, ADS8584S analogue to digital converter (ADC), and sync times to the control computer. The camera trigger frequency follows a predetermined sequence and is iterated through a command in the Gleeble Script Language (GSL) program, which controls the Gleeble 3800, thus allowing for event-based frame-rate changes. State and trigger light indications are incorporated onto the board for convenience during operation (see Appendix A2 for details).

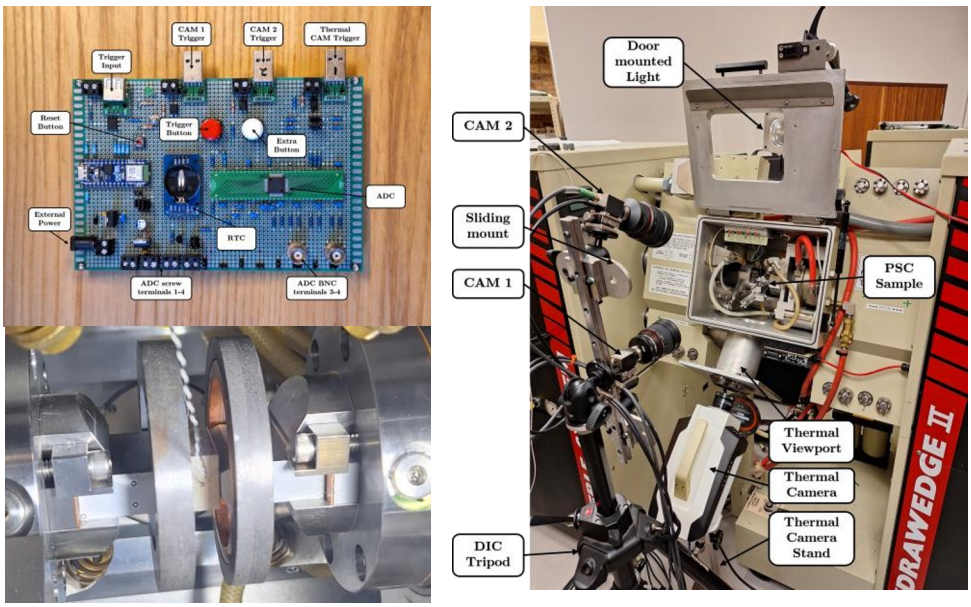

Fig. 1. a) DIC capture circuit board, b) closeup of PSC sample loaded into the Gleeble 3800, and c) overview of full hardware setup.

2.2 Operation of DIC capture and graphic user interface (GUI)

To begin operating, the user first runs the gui.py Python program to enter the DIC Capture system. The user selects a main working folder for the project they are working on. Various configurations can be saved as config.json files and loaded from this folder for future tests. This ensures that all test variables are consistent between tests and shortens the setup time. Next, a graphical user interface (GUI) (Fig. 2) was developed to allow the user to operate the system in two run modes. Firstly, the "Run Test Mode" is used when setting up an experiment; whereby the cameras are configured to operate as fast as possible with functions such as zoom, show histogram, adjust exposure and display live ADC values are all enabled. Only Camera 1 and 2 parameters required configuration before initializing the "Run Test Mode" process. Secondly, the "Run Record Mode" saves all incoming data and is used when running an experiment for results, all additional features are disabled to prioritize compute resources and maximize stability. A unique "Test ID" must be set before entering record mode, a test folder is created with the Test ID name, where all data relevant to the current test are stored. By creating a new folder for every test, the possibility of overwriting previous data is eliminated and if the file name already exists, the program does not proceed and prompts the user to change the file name. A default version of the test ID is also loaded with the selected config file (see Appendix A3 for detailed operating instructions and Appendix A4 for details on DAQ data structures).

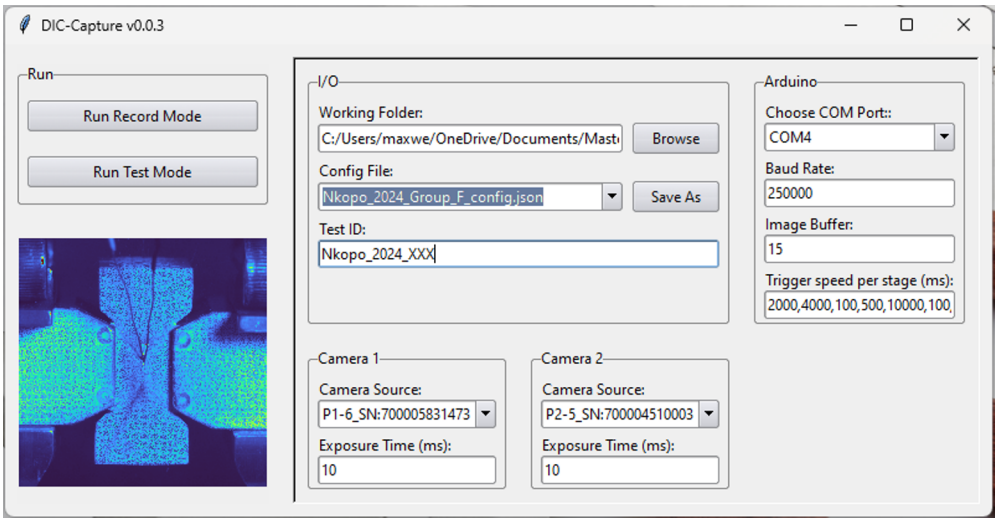

Fig. 2. DIC Capture GUI.

2.3 Synchronization methodology

The algorithm used in the *'post_processor_syncronization.py'* program adaptively reads the data from the Gleeble output file. The Gleeble time-stamps for the rising edges of the DIC.Trigger dataset are compared to the *fps_change_time* in the Arduino output file. The average difference between the two is used to shift the *test_run_time* of the DIC data. The DIC data can then be interpolated into the Gleeble data. The user has the option to generate a Matplotlib graph of the *Gleeble Force* and *DIC ADC Force* vs *time* (Fig. 3a) to ensure that the synchronization was successful (Fig 3b). The DIC Capture ADC values are shown in red when an image is captured and blue for the ADC subsampling, the black plot is the force output from the Gleeble data. An approximate conversion from ADC volts to force was used

for demonstration purposes, typically the user should verify that the conversion ratio is correct for their specific test setup. The Gleeble data was averaged between samples when sampling at lower frequencies, which can create discrepancies between the DIC Capture ADC values and Gleeble data if there is a significant difference in sampling frequency between the two systems.

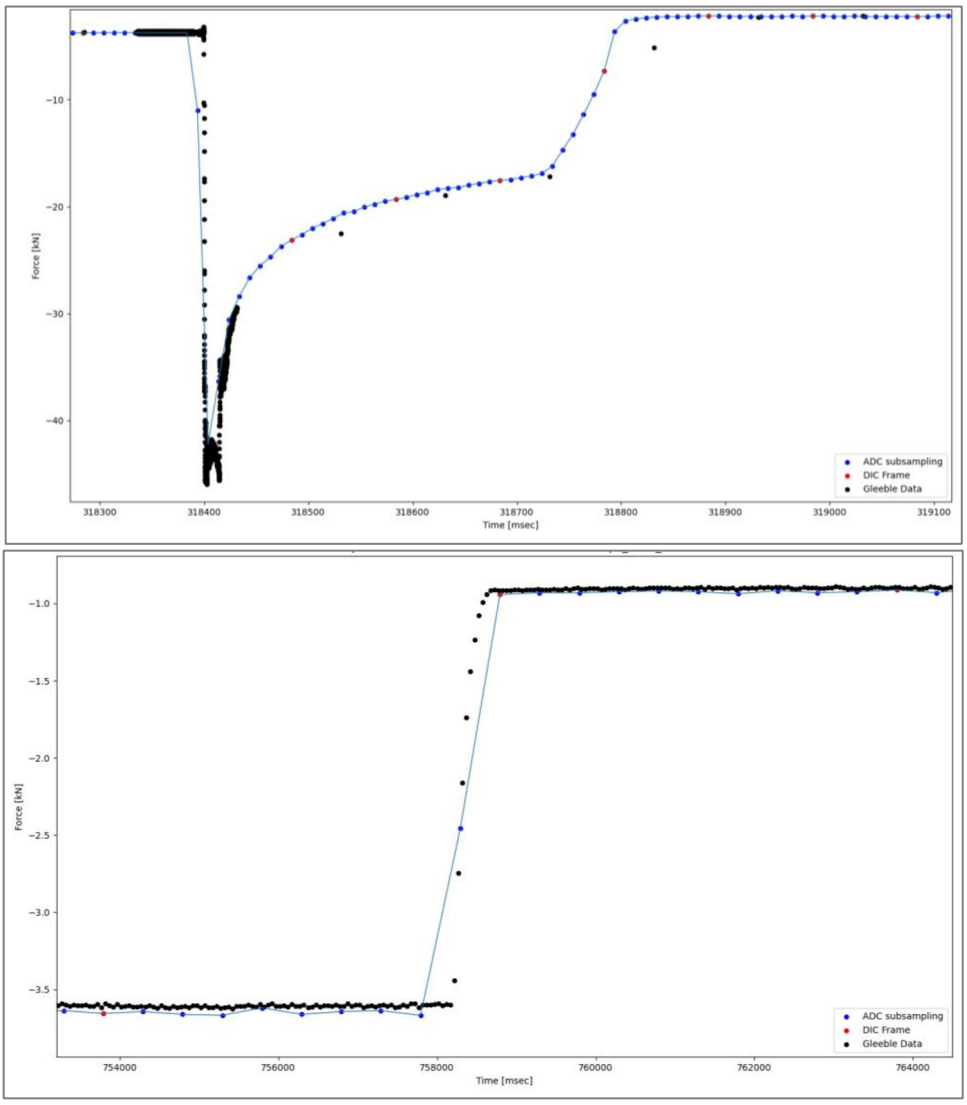

Fig. 3. Graph generated during synchronization showing a) PSC Hit and b) DIC capture data and Gleeble data when turning air ram off.

2.4 Reformatting thermal images for MatchID software

Thermal images were captured using the Telops FAST M350 thermal camera with the Reveal IR V1.10.1 software package. Each sequence was saved in a proprietary .hcc project format and images can be exported as raw .tiff files. It must be noted that the older versions of the Reveal IR software cannot export in this format so one should update the software if using

an old computer. The raw .tiff files have floating point pixel values which represent the temperature in Celsius for each pixel. This is different from the usual image convention of representing pixels on an integer scale between 0 and 2^n-1. This floating-point format is convenient for temperature data as no further conversions are required; however, this format is not widely used and cannot be displayed on most software. When spatially calibrating the thermal camera with the primary DIC camera, the thermal image must have the same format as the DIC camera (see Appendix A5 for calibration details). Once calibration is completed in MatchID using the reformatted images, the thermal images used for the *Thermal Analysis* module in MatchID need to be formatted specifically to work with the module. These images are saved as pseudo text files with the extension of .thermal.tiff, with the required structure as shown in Fig. 4 (see Appendix A6 for conversion).

```
***MatchID Thermal
<Width>=<2448>
<Heigth>=<2048>
<Unit>=<1>
<Matrix>
24.07;24.07;24.07;24.07;24.07
24.07;24.07;24.07;24.07;24.07
24.07;24.07;24.07;24.07;24.07
24.06;24.06;24.06;24.06;24.06
24.05;24.05;24.05;24.05;24.05
24.04;24.04;24.04;24.04;24.04
```

Fig. 4. MatchID .thermal.tiff formatting.

3 Case study

The reliability of the system was assessed through real world DIC tests on the Gleeble 3800. The objective was to generate data that proved the DIC Capture system integrated with the Gleeble 3800 and MatchID software and produced useful raw DIC data for further processing and analysis.

3.1 Testing procedure

The testing for the case study included multi-deformation PSC tests on aluminium alloy AA3104 with deformation temperatures of 360 °C, 330 °C and 300 °C; at strain rates of 30/s, 50/s and 100/s for true strains of 0.3, 0.32 and 0.4, respectively. 32 tests were performed with simultaneous DIC and thermal imaging integration.

3.2 Step-by-step test procedure

In order to provide a representative overview of the complete testing procedure, an instructional step-by-step guide is presented below. This guide focuses on the steps using the DIC Capture system, and thus does not present extraneous details on standard operation of the Gleeble 3800.

1. The sample is loaded in the Gleeble 3800 according to the Gleeble 3800 SOP, with the specific test setup for this validation run shown in Figure 9.

2. The gui.py program is run. The experiment's working folder is selected and the config file for the testing group is loaded. The test ID is inputted, and the rest of the fields are confirmed to be correct.

3. Within the Reveal IR software, the thermal camera is configured to Falling Edge hardware triggering and the output directory is set to a file with the same name as the test ID. Check that the record mode is in radiometric temperature and that the thermal cameras frame rate is set to a value which exceeds the maximum frame rate that will be outputted from the DIC Capture system. The 'HDD' button is then clicked, indicating that all images will be recorded from this point onwards.

4. The 'Run Record Mode' button was activated on the DIC Capture GUI. This initialised the Arduino, DIC Cameras and notified the user when that the system was ready to run, as seen in Fig. 5.

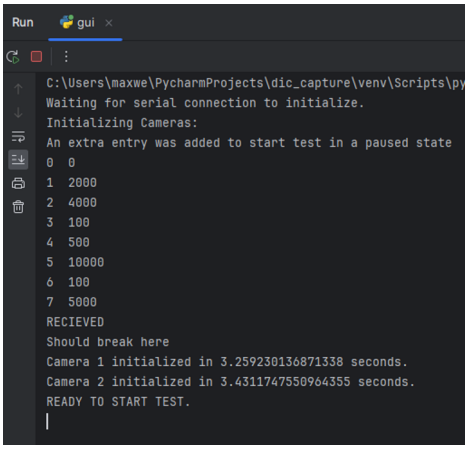

Fig. 5. Pre-test consol output confirming test was initialized correctly.

5. The GSL program is then run as normal on the Gleeble 3800. When the FPS change sequence is called in the GSL (see Figure 2) for the first time, DIC acquisition was started. The DIC images were displayed on the control computers screen as they are saved to storage. The thermal images were displayed on the Reveal IR software.

6. At the end of the test, when the status LED on the Arduino was observed to be red, the DIC capture software had stopped and the sequence was stopped on Reveal IR. The thermal images were then exported in .tiff format to the 'Camera 3' folder within the main test folder for the current test. The Gleeble output files can be manually transferred to the 'Raw data' folder; otherwise, this can be done when running the synchronisation program.

7. The thermal images are prepared by running the 'telops_to_matchid_reformat.py' script. The user was prompted to select the test folder from which the images are uploaded. Fig 6 shows a single frame from the various cameras after being processed.

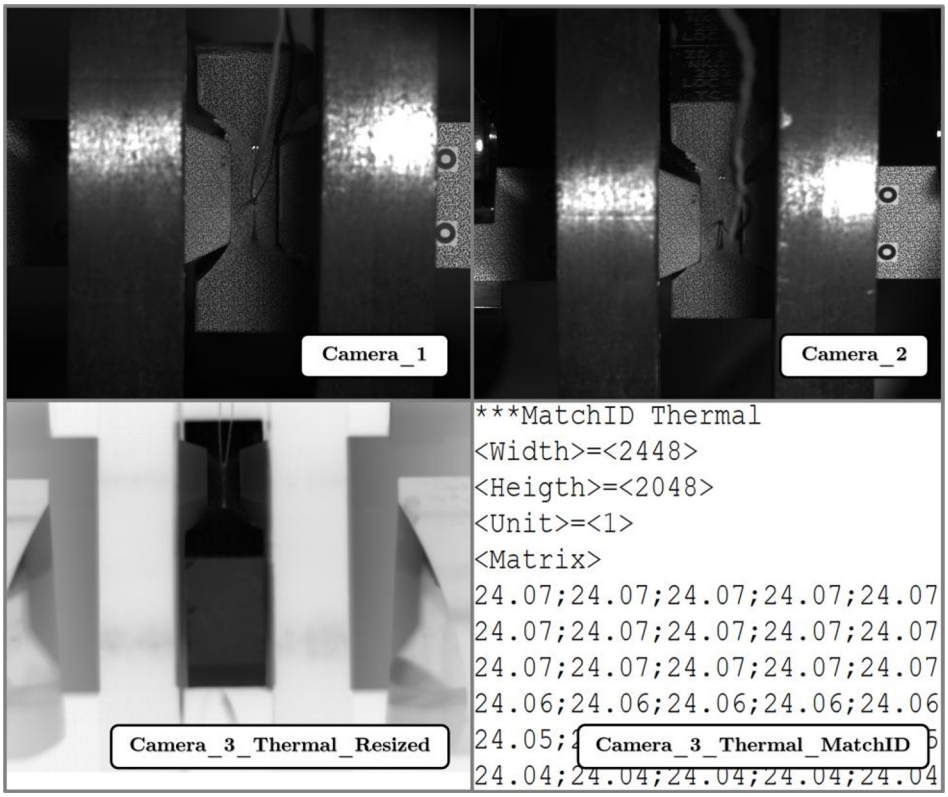

Fig. 6. Comparison of the different camera outputs for the same frame number.

8. The 'post_processor_syncronization.py' program was then run, with the output being two .csv files. One was named 'SyncedGleebleData_testID' and contains all the Gleeble and DIC data combined into one dataset with the Gleeble time used as reference. The other file was named 'MatchID_AcquisitionFile_testID' and is formatted to be compatible with MatchID, this file only contains the data associated with each image. These files are saved in the 'Synced_Data' folder within the main test folder.

9. At this stage, the Gleeble, DIC images, DIC ADC data and thermal images are all temporally synchronized and are formatted correctly to be used in MatchID or any open source DIC software provided that the user adheres to the formatting conventions of the software.

10. The test files were then transferred to the computer where MatchID is installed. Camera 1 and Camera 2 were first calibrated, followed by Camera 1 and Camera 3, Camera 3 is the thermal camera, see Figure 12. The images in the 'Camera_3_Thermal_Resized' folder were used for the calibration process as the calibration process required both images to be in the same format and size.

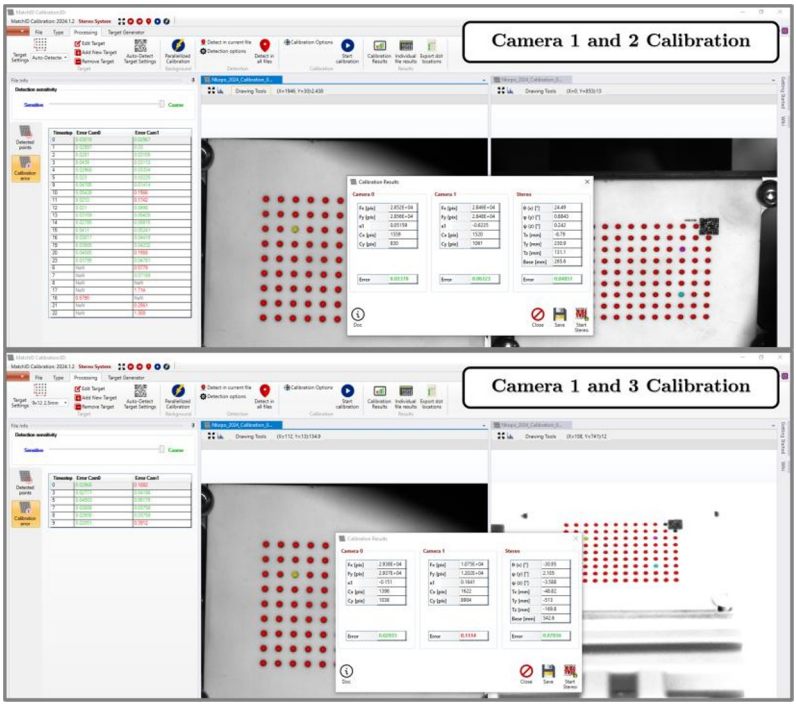

Fig. 9. MatchID calibration of camera 1 and 2 (top), as well as camera 1 and 3 (bottom).

11. MatchID Stereo was then opened, where images from Cameras 1 and 2 were imported with a reference image set from camera 1. A region of interest was then selected. The user was prompted to set any other DIC pre-processing parameters at this stage. To import the DAQ data for each image, the user engaged the "Acquision Data" button and selected the 'MatchID_AcquisitionFile_testID' within the current working test folder. Figure 13 shows an example of the preview of the data, which should be checked before proceeding.

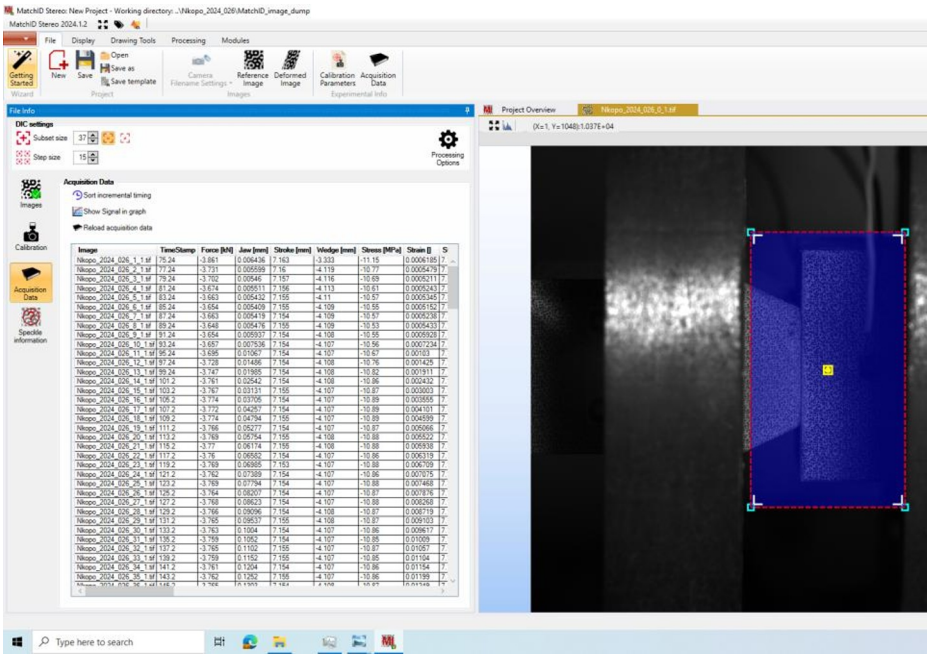

Fig. 10. Preview of DAQ data generated by DIC Capture and imported into MatchID.

12. After processing the DIC data, the "Temperature Import" module was run. The "Manual Upload" was used to select all the ".thermal.tiff images" in the dataset, as shown in Fig. 11. The calibration file for Cameras 1 and 3 was used for this stage when selecting "Calibration Parameters".

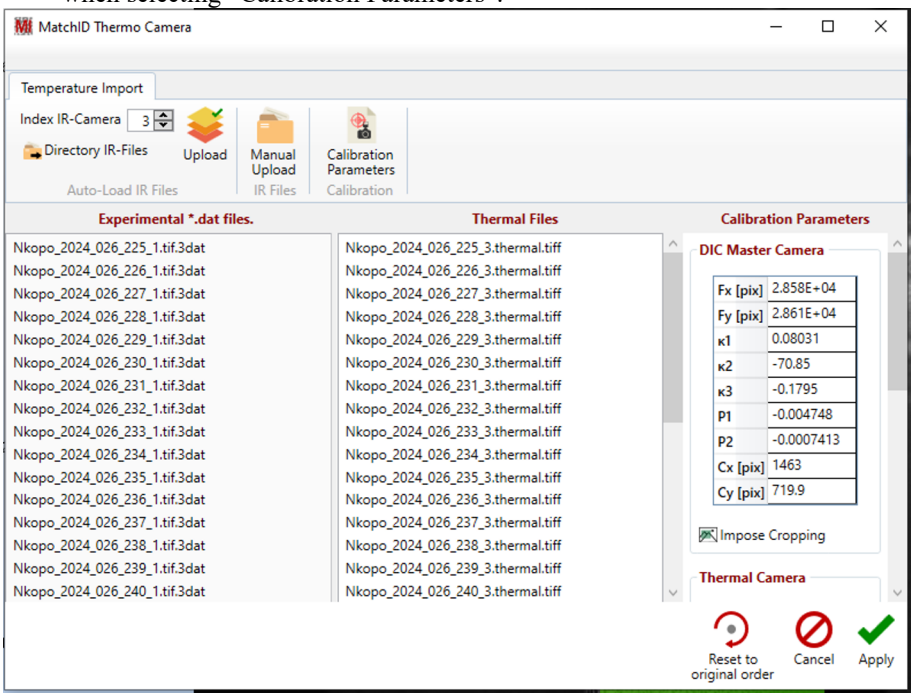

Fig. 11. Import of thermal data into MatchID temperature import module.

13. Fig. 12 demonstrates the image correlation, where the global vertical displacement, of the sample surface relative to the reference image, is shown during the quenching process.

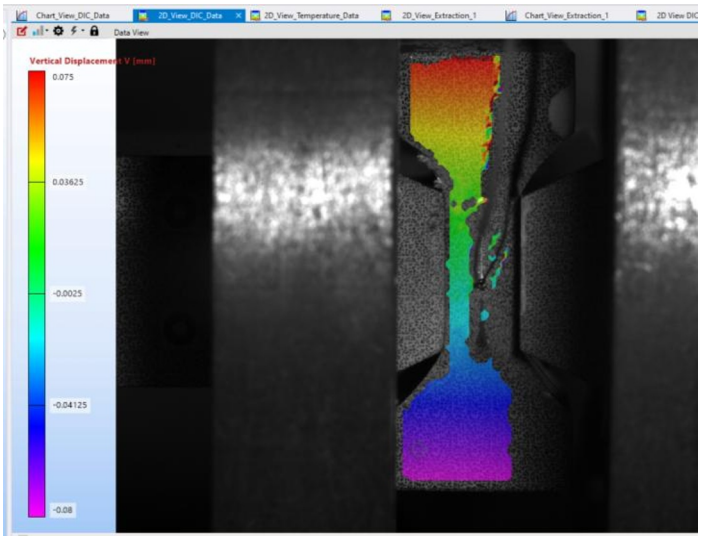

Fig. 12. Vertical displacement of PSC sample during quenching.

4 Discussion

Beyond the case study presented above for PSC testing of aluminium, the DIC Capture system with integrated thermal mapping had also been used for high temperature (850 °C) tensile testing of additively manufactured Inconel 718 [7, 14]. From this example, it was demonstrated that a large field of view could be achieved with simultaneous overlays of both the DIC and thermal information as shown in Fig. 13.

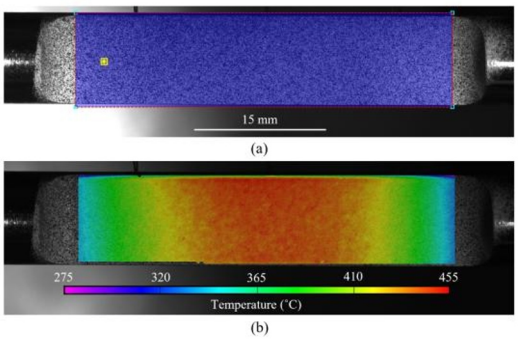

Fig. 13. a) Specimen DIC field of view and b) infrared overlay onto the same field of view. Used with permission under CC-BY-4 [14].

Based on the use of the system over a range of materials, different mechanical testing configurations, and temperature ranges, several key methodological necessities can be highlighted for the efficacy of linking the systems using the presented DIC Capture system. Firstly, one system must be used as the reference system. Within this context, the reference system must be capable of accurate time keeping with hardware interrupts, at least one

analogue input, one digital output and a means to save or transmit basic data. The reference time of received and sent triggers is recorded for synchronisation. Secondly, to sync with a testing machine's (e.g., Gleeble 3800) output data, the time of at least one, ideally two, events must be known relative to the machines output data. For testing machines that can provide an analogue voltage output for signals, the ADC on the reference system can be used as an alternative. Both sync pulses and ADC readings can be used if desired and is recommended for complex systems, such as the Gleeble 3800, as redundant synchronisation methods improve the robustness of the testing system. Thirdly, any camera that can receive a hardware trigger can be used with the DIC Capture system. This is achieved by identifying the total count and time of triggers sent from the reference system and then post processing the images into a standardised format. This approach was chosen to ensure that any camera can be used with the DIC Capture system in at least a basic, but functional, manner. One pitfall of this approach is that a single frame drop can potentially desynchronise the camera from the system. A mitigation technique used for this problem is to record the camera's local time with the image data where possible. If a discrepancy is noticed, the user can resynchronise the camera data. With this potential problem known, additional care was taken when writing the image saving algorithm in the DIC Capture python software which has resulted in a reliable and versatile system. Lastly, formatting and synchronisation of data should be done after the test is completed, not during the test. The unprocessed recorded data is saved to storage as soon as possible. This approach greatly simplifies and improves the performance of the DIC Capture system.

5 Conclusion

The open-source DIC capture system was capable of producing high quality temporally synchronised DIC and thermal data sets and integrated with the Gleeble 3800 for high temperature applications. The DIC capture system was successfully tested using several different case studies over a range of testing conditions.

References

[1] Gleeble 3800 GTC. (accessed https://gleeble.com/products/gleeble-systems/gleeble-3800.html).

[2] J. Yang, S. Sung, C. Chen, and A. Tan, Study of microstructural and mechanical properties of weld heat affected zones of 2024-T3 aluminium using Gleeble simulation. Mater. Sci. Technol. **27**, 357-365 (2011)

[3] M. van Rooyen, T. H. Becker, J. E. Westraadt, and G. Marx, J, Measurement of creep deformation of ex-service 12% Cr steel using digital image correlation. Strain Anal. Eng. Des. **55**, 71-85 (2020)

[4] M. van Rooyen and T. H. Becker, High-temperature tensile property measurements using digital image correlation over a non-uniform temperature field. J. Strain Anal. Eng. Des. **53**, 117-129 (2018)

[5] J. Qin, D. Racine, K. Liu, and X. Chen, *Strain-controlled thermo-mechanical fatigue testing of aluminum alloys using the Gleeble 3800 system* in Proceedings of the 16th International Aluminum Alloys Conference (ICAA 16), Canadian Institute of Mining, Metallurgy & Petroleum, Montreal, QC, Canada, 17-21, (2018)

[6] Y. Gajalappa, A. Krishnaiah, K. B. Kumar, K. K. Saxena, and P. Goyal, Flow behaviour kinetics of Inconel 600 superalloy under hot deformation using gleeble 3800. Mater. Today Proc. **45**, 5320-5322 (2021)

[7] M. Blackwell, M. Vos, S. George, M. Neaves, and T. Becker, Temperature Dependant Mechanical Property Characterisation Using Digital Image Correlation and Infrared Thermography. R&D J. **40**, 27-32 (2024)

[8] B. Pan, K. Qian, H. Xie, and A. Asundi, Robust full-field measurement considering rotation using digital image correlation. Meas. Sci. Technol. **20**, 062001 (2009)

[9] E. M. Jones and M. A. Iadicola, A Good Practices Guide for Digital Image Correlation. Int. Dig. Image Correl. Soc. **10**, 1-110 (2018)

[10] V. Belloni, R. Ravanelli, A. Nascetti, M. Di Rita, D. Mattei, and M. Crespi, Digital image correlation from commercial to FOS software: a mature technique for full-field displacement measurements. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. **42**, 91-95 (2018)

[11] D. Atkinson and T. Becker, A 117 line 2D digital image correlation code written in MATLAB. Remote Sens. **12**, 2906 (2020)

[12] L. Yu and B. Pan, Overview of high-temperature deformation measurement using digital image correlation. Exp. Mech. **61**, 1121-1142 (2021)

[13] MatchID. (accessed https://www.matchid.eu/).

[14] M. A. Blackwell, *High-temperature mechanical property characterisation of additively manufactured Inconel 718*. MSc thesis, Stellenbosch University (2024)

Appendix

Appendix A1. ADC selection

The ADS8584S from Texas Instruments was chosen for this project. The ADS8584S met the following criteria:

1. ±10V range with simultaneous sampling on 4 channels. There are seldom scenarios where more than two ADC inputs are needed; however, having the 4 inputs would be useful for more complex experiments and the additional cost to include the extra 2 channels was negligible.
2. 16-Bit resolution should ensure that the resolution of the ADC is not the limiting factor of the system accuracy.
3. Adequate sampling frequency, with a maximum of 330 thousand samples per second per channel, multiple samples can be averaged to produce a more stable reading.
4. Cost effective and readily available from www.digikey.com (part number 296-50900-1-ND) who ship worldwide.

Appendix A2. State and trigger indication light.

When using Version 2 of the trigger board, an oscilloscope was used to track the frame triggers to monitor for any problems. The inability to easily monitor the current state of the Arduino created opportunity for confusion or mistakes when testing. Version 3 has a green LED wired to the camera trigger signal that shows when each camera is triggered. There is a

RGB led built into the Arduino Nano ESP32 that is utilized to indicate that the board is in one of the following states:

1. Solid blue: Arduino setup code is running.
2. Flashing blue: ready to receive frame rates from the control computer.
3. Solid cyan: paused and waiting for first trigger signal.
4. Solid green: Cameras are being triggered.
5. Solid yellow: Camera triggering has been paused (occurs when a zero millisecond period is sent), but the test is not over.
6. Solid red: Test is over, indicating that the any capturing software attached to the board should be stopped, saved or reset. The Arduino can then be reset for the next test.

Appendix A3. Overview of the DIC capture system

In the Arduino section of the GUI, a list of available COM port devices is automatically populated. The user is responsible for selecting the COM port associated with the triggering device; in practice though, there is seldom more than one COM port option. The baud rate should not be changed from 250 000 unless a different microcontroller is used for the triggering device. This field is automatically populated.

The “Image buffer” indicates that every n’th image received from the cameras will be displayed on the computer.

The “Trigger speed per stage” is one of the main reasons for embarking on this project. For tests where events happen at different time scales, selecting one frame rate to adequately capture DIC data is often impractical. By inputting a string of comma delaminated integer values, which represent the triggering period in milliseconds, the DIC capture system will iterate through these values each time a FPS change signal is received. This FPS change signal is a 5 V rising edge signal sent from the Gleeble, either by calling the DIC.trigger or Quench 4 function in the GSL file being used for the current Gleeble test. An example of this code is seen in Fig. A1, if the user does not have the DIC module from Gleeble, they can simply connect it to the Quench 4 output from the Gleeble and replace the DIC.trigger with a Quench 4 on/off pulse. Note that Quench 4 has a minimum cycle time of approximately 500 ms. When a FPS change pulse is received by the DIC Capture system, a 30 ms delay is included to ensure that the cameras are not accidentally double triggered during a FPS change. This delay is not present for the first FPS pulse that it received in order to maintain synchronous starting times. It is important to sample at a high frequency when calling the DIC.trigger as the recorded DIC.trigger data is used to synchronize the Gleeble output data with the DIC capture data. It was found that a 100 ms delay before, and a 130 ms delay after the DIC.trigger is required to ensure that the rising edge is captured at 1000 Hz. Shorter delays may be possible; however, this is challenging to optimise due to the limitations of the GSL system of the Gleeble 3800.

```
//=============================================(/1)
// Start of soak
// Frame Period: 4000ms
repeat 1
    sample at 1000Hz
    sync
    delay 100msec
    sync
    set DIC.trigger to on
    sync
    delay 130msec
    sync
    sample at 20Hz
    sync
end
//=============================================(1/)
```

Fig. A1. FPS change code in GSL

The camera 1 and 2 input boxes are used to select the USB port ID for each camera. The drop-down box selection is automatically populated by using the Baumer neoAPI to find the port ID and associated serial number for each camera. It is generally recommended to make camera 1 the reference camera as this makes the calibration process easier during post processing. Users can either match the serial numbers on the camera to those shown in the GUI or unplug one camera to identify which camera is associated with its respective port.

Appendix A4. DAQ data structures

The formatting of the serial data from the Arduino has changed from the previous version. The data column now has the following names:

"trigger_frame_count,
adc_frame_count,
test_run_time,
fps_change_count,
fps_change_time,
adc.ch1_volts,
adc.ch2_volts,
adc.ch3_volts,
adc.ch4_volts,
data_delta".

A description of each category is listed below:

1. *trigger_frame_count*: a count value starting at 0 which increments each time a trigger pulse is sent to the cameras. *trigger_frame_count* is equal to zero for the first image.
2. *adc_frame_count:* similar to *trigger_frame_count*. A count value starting at 0 which increments each time the ADC records an input voltage. Currently the system is set to record 10 ADC records for each image which is acquired. So, if the triggering frequency of the cameras is 10 Hz, the ADC will record and save data at 100 Hz.
3. *test_run_time:* the time in milliseconds since the first image was taken.
4. *fps_change_count:* A count value which is incremented each time a FPS change signal is received. Mainly used for data management purposes.

5. *fps_change_time:* The exact *test_run_time* when an external FPS change signal is received. This is used for syncing data with external systems. For the Gleeble, the DIC.Trigger or Quench 4 data can be acquired to match Gleeble data with the DIC data. In general; any system where the time of a rising edge signal is known, can be temporally synchronized to the DIC Capture system. It is worth noting that only a start and stop signal are required if no FPS changes are required during the test.
6. *adc.chX_volts:* Output from each ±10V ADC channel. There are 4 channels which can be use simultaneously. It is left to the user to convert the voltage into the equivalent unit for their sensor output. It is the user's responsibility to calibrate their system. This can be done by inputting known voltages and reading what is recorded in the *adc.chX_volts,* a calibration curve can then be generated from the differences in these values.
7. *data_delta:* Used to track if the serial communication speed is keeping up with the data generation. This should always be zero. If it increases, one should decrease the amount of ADC records taken between frames. In the Current configuration, 33 FPS is the maximum framerate with a 10:1 ratio of ADC to frame triggering. 33 FPS is the maximum continuous speed for the cameras we generally use (Baumer VCXU50M) at 12-bit pixel depth, hence the choice of 10:1 ratio. On board storage solutions could be implemented in future to increase maximum data rate.

Appendix A5. Calibration of thermal camera with primary DIC camera

Calibration is done in the *'telops_to_matchid_reformat.py'* program by performing the following operations to the thermal image:

1. Cycle through each image in the user selected directory, loading each image into a NumPy array at the start of the reformatting loop.
2. Normalise the pixel values of the image to a scale between 0 and 255. 8-bit format was used as accurate bit depth is not required for calibration.
3. The image is resized to the same resolution as the DIC image. If the thermal and DIC images have a different aspect ratio; a black bar is added to either the right or bottom side of the image, depending on the aspect ratio. A linear interpolation is used during the upscaling of the thermal image to the DIC image resolution.
4. The pixel values are rounded to the nearest integer value and converted to the UINT-8 format.
5. The array is saved as a .tiff image in a separate '*Camera_3_Thermal_Resized'* folder with an image naming convention which matches the DIC image convention, where the camera ID is 3.

Appendix A6. Conversion of .tiff images into the MatchID compatible thermal.tiff format

The process for converting the exported .tiff images into the MatchID compatible thermal.tiff format using the '*telops_to_matchid_reformat.py'* program is summarized as:

1. Build a header string consisting of the first 5 lines shown in Fig. A2. Note the spelling of 'Heigth', at the time of writing, the typo must be included. The image width and height are defined at the start of the '*telops_to_matchid_reformat.py'* program and should match the resolution of the DIC cameras used for the experiment.

2. Cycle through each image in the user selected directory, loading each image into a NumPy array at the start of the reformatting loop.
3. The image is resized to the same resolution as the DIC image. If the thermal and DIC images have different aspect ratios; a black bar is added to either the right or bottom side of the image, depending on the aspect ratio. A linear interpolation is used during the upscaling of the image.
4. Use the NumPy '*savetxt()*' function to save this image with the format shown in Fig where the *output_match_ID_path* is the file name of the image ending in '.thermal.tiff'. These images are saved to a '*Camera_3_Thermal_MatchID'* folder withing the main folder for the test. It is worth noting that in the 'fmt' section the number of decimals can be changed; in Fig, the temperatures are rounded to whole numbers, but this can be increased if needed. Adding more decimal places increases the file size of each image, and the file sizes are already sizable due to the unique format, so this should be considered if there are many images per test.

```
np.savetxt(output_match_ID_path,
           boarder_img,
           fmt='%1.0f',
           delimiter=';',
           newline='\n',
           header=heading_string,
           comments='',
           encoding=None)
```

Fig. A2. NumPy savetxt format for MatchID .thermal.tiff files.