remote sensing

Article

A 117 Line 2D Digital Image Correlation Code Written in MATLAB

Devan Atkinson and Thorsten Becker *

Department of Mechanical and Mechatronic Engineering, Stellenbosch University, Corner of Banghoek and Joubert Street, Stellenbosch, Western Cape 7599, South Africa; 17732913@sun.ac.za
* Correspondence: tbecker@sun.ac.za

Received: 22 July 2020; Accepted: 3 September 2020; Published: 8 September 2020

Abstract: Digital Image Correlation (DIC) has become a popular tool in many fields to determine the displacements and deformations experienced by an object from images captured of the object. Although there are several publications which explain DIC in its entirety while still catering to newcomers to the concept, these publications neglect to discuss how the theory presented is implemented in practice. This gap in literature, which this paper aims to address, makes it difficult to gain a working knowledge of DIC, which is necessary in order to contribute towards its development. The paper attempts to address this by presenting the theory of a 2D, subset-based DIC framework that is predominantly consistent with state-of-the-art techniques, and discussing its implementation as a modular MATLAB code. The correlation aspect of this code is validated, showing that it performs on par with well-established DIC algorithms and thus is sufficiently reliable for practical use. This paper, therefore, serves as an educational resource to bridge the gap between the theory of DIC and its practical implementation. Furthermore, although the code is designed as an educational resource, its validation combined with its modularity makes it attractive as a starting point to develop the capabilities of DIC.

Keywords: digital image correlation; two-dimensional; subset based; education; MATLAB code

1. Introduction

Digital image correlation (DIC) determines the displacements and deformations at multiple points spanning the surface of an object (full-field displacements and deformations) from images captured of the object. It is type of a full-field, non-contact optical technique and these techniques are categorised as either interferometric or non-interferometric. The interferometric techniques, such as Electronic Speckle Pattern Interferometry and Moiré Interferometry, require a coherent light source and need to be isolated from vibrations [1]. As such, their utilisation is in the confines of a laboratory. In contrast, non-interferometric techniques, DIC and the grid method require simple incoherent light and are more robust with regards to ambient vibrations and light variations [2]. Thus, non-interferometric techniques are more attractive due to their less stringent requirements and are mostly used in open literature. DIC allows for a more straightforward setup compared to the grid method as it only requires a random, irregular pattern on the surface of the object instead of a regular grid.

These advantages of DIC over other full-field, non-contact optical techniques, along with the decreasing cost and increasing performance of digital cameras, has led to widespread use of DIC in various fields. Some applications of DIC include: (i) performing human pulse monitoring [3,4]; (ii) analysing the stick-slip behaviour of tyre tread [5]; (iii) determining the mechanical properties of biological tissue [6–8]; (iv) in situ health monitoring of structures and components [9–11]; (v) analysing vibration of components [12,13]; and (vi) remote sensing applications [14–17]. However, DIC has received the most attention, and thus development, for applications in experimental solid mechanics. As such, this paper will predominantly focus on DIC in the context of experimental solid mechanics applications.

In the field of experimental solid mechanics, measuring the displacement and deformation experienced by a specimen, as a result of an applied load, is essential to quantify its mechanical properties. As such, DIC is advantageous for three reasons: Firstly, its full-field nature allows more complex constitutive equations to be used to determine more than one material property at a time, using methods such as the virtual fields method [18–20] and the finite element model updating method [21]. Secondly, the non-contact nature of DIC avoids altering mechanical properties of the materials being tested, such as in the case of determining the material properties of biological tissue [6–8] and hyper-elastic materials [22]. Lastly, DIC allows the specimen to be exposed to harsh environments, such as high-temperature applications, while still being able to take measurements, provided the specimen is visible [23].

When DIC was first introduced by Peters and Ranson in 1982 [24], it used a simple cross-correlation criterion with a zero-order shape function (SF) and could not account for the deformation of the specimen or variations in ambient light. Between 1983 and 1989, Sutton and his colleagues improved the technique by introducing the first-order SF [25], the normalised cross-correlation criterion which is more robust against light variations [26], the Newton–Raphson (NR) optimisation method [27] and bi-cubic b-spline interpolation [28]. The two-dimensional (2D) DIC technique was extended to three dimensions (3D or stereovision DIC) in 1993 by Luo et al. [29] and to digital volume correlation (DVC) in 1999 by Bay et al. [30] using X-Ray tomography-computed images.

The most significant contributions to the current state-of-the-art DIC technique, as identified by Pan [2], occurred during the 21st century. In 2000, Schreier et al. [31] proved that bi-quintic b-spline interpolation is the best interpolation method for accurate sub-pixel displacements. In the same year, Lu and Cary [32] introduced the second-order SF to account for more complex deformations. In 2004, Baker and Matthews [33] proposed the inverse compositional Gauss–Newton (IC-GN) optimisation method using the sum of squared difference correlation criterion which is more efficient than the NR method. However, Tong showed in 2005 [34] that the zero-mean normalised sum of squared difference (ZNSSD) correlation criterion is the most reliable and so Pan et al. [35] adapted the IC-GN method to use the ZNSSD criterion in 2013. Finally, Gao et al. [36] introduced the second-order SF to the IC-GN method in 2015. The IC-GN method is considered to be the state-of-the-art optimisation method because it has been shown to be theoretically equivalent to the NR method [33] while offering improved accuracy, robustness to noise and computational efficiency in practice [37].

The DIC process is complicated, comprising of several intricate elements, including correlation, camera calibration, transformation of displacements between the device and real-world coordinates and strain computation. Successful application of DIC requires an understanding of all these elements and thus newcomers to the field need to overcome a difficult learning curve. To this end, there are several papers which give a comprehensive breakdown of the theory involved in the DIC process, such as the papers by Pan et al. [35], Gao et al. [36] and Blaber et al. [38]. However, in order to contribute towards the development of DIC, a deep understanding of the DIC process and its elements is required. It is incredibly time-consuming to gain this working knowledge due to a lack of publications that directly bridge the gap between the theory and its implementation in code. More specifically, papers either do not provide code that details the implementation of the theory in practice [35,36] or the code that they provide is too complex to be beneficial as a learning resource [38].

This paper aims to bridge the gap between the theory and implementation of DIC. It does this by firstly presenting the theory for a 2D, subset based DIC framework that is predominantly consistent with current state-of-the-art practices. Thereafter the implementation of the theory of the framework as the provided 117 line MATLAB code is discussed. Lastly the correlation aspect of the code is validated using the DIC Challenge image sets documented by Reu et al. [39]. More specifically, its results are discussed in parallel with those obtained using either the commercial software package by LaVision

(Davis) and the open-source software Ncorr [38] or, to results documented in the DIC Challenge paper [39], in order to draw conclusions.

The framework, referred to as the ADIC2D framework, is implemented using MATLAB because its simple syntax does not distract the reader from the mathematics of the code. Additionally, its built-in functions are used to simplify the code and improve its efficiency. The code is modular, allowing readers to progressively build up their understanding of the code so that recognising the connection between the theory and code is straightforward. Moreover, this modularity allows for rapid adaption of the code thereby encouraging readers to develop the capabilities of DIC.

2. Framework Theory

DIC consists of four processes: calibration, correlation, displacement transformation and strain computation. Calibration involves determining the parameters of the camera model which relates the location of a point on an object in the real world to the location of the corresponding point in an image taken of the object. Correlation calculates how portions of the object, captured in the image set, displace throughout the image set. Displacement transformation then uses the parameters determined by calibration to transform the pixel displacements determined by correlation to metric displacements in the real world. Finally strain computation determines the strain fields experienced by the specimen from the displacement fields.

2.1. Calibration

Calibration determines the parameters of the camera model. ADIC2D uses the pinhole camera model to transform the location of a point in the real world to the idealised location of the point in the image. Then, a radial distortion model is used to relate the idealised location of this point to its actual distorted location, as illustrated in Figure 1.

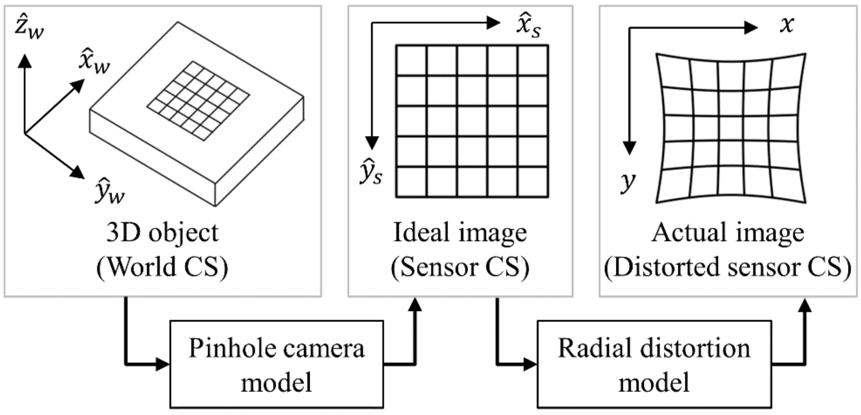


Figure 1. Schematic diagram illustrating how the camera model is comprised of the pinhole camera model and radial distortion model.

2.1.1. Homogeneous Coordinates

The pinhole camera model works with homogeneous coordinates as these allow rotation, translation, scaling and perspective projection to be applied using matrix multiplication. An n-element vector, which represents a point in n-dimensional space, is converted to homogeneous coordinates by appending a scaling variable of unity to the end of the vector. Converting back from homogeneous coordinates involves dividing each element of the vector by the last element, the scaling variable, before removing the last element. Homogeneous coordinate vectors are indicated by underlining the variable name. For more information on homogeneous coordinates, refer to the work of Bloomenthal and Rokne [40].

2.1.2. Pinhole Camera Model

The pinhole camera model relates the location of a point in the world coordinate system (CS) to its corresponding idealised location in the sensor CS. The 3D world CS is defined such that its x-y plane is coincident with the surface of the specimen under consideration: 2D DIC is limited to determining displacements that occur within this x-y plane. The 2D sensor CS is defined such that its x-y plane is coincident with the plane of the charge-coupled device which captures light rays incident upon its surface as an image.

Let the homogeneous coordinates in the world and sensor CS be $\underline{\hat{\boldsymbol{x}}}_w = \begin{bmatrix} \hat{x}_w & \hat{y}_w & \hat{z}_w & 1 \end{bmatrix}^T$ and $\underline{\hat{\boldsymbol{x}}}_s = \begin{bmatrix} \hat{x}_s & \hat{y}_s & 1 \end{bmatrix}^T$, respectively. Note that the circumflex indicates that the coordinates are ideal (undistorted). The pinhole camera model is given as [41]:

$$\alpha\underline{\hat{\boldsymbol{x}}}_s = \alpha\begin{bmatrix} \hat{x}_s \\ \hat{y}_s \\ 1 \end{bmatrix} = \begin{bmatrix} \xi_x & c_s & c_x \\ 0 & \xi_y & c_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{bmatrix}\begin{bmatrix} \hat{x}_w \\ \hat{y}_w \\ \hat{z}_w \\ 1 \end{bmatrix} = \boldsymbol{K}\boldsymbol{V}\underline{\hat{\boldsymbol{x}}}_w, \tag{1}$$

where matrices $\boldsymbol{V}$ and $\boldsymbol{K}$ contain the extrinsic and intrinsic camera parameters respectively. The extrinsic camera parameters define a rotation matrix $\boldsymbol{R}$ and a translation vector $\boldsymbol{T}$ which define the position and orientation of the world CS relative to the position and orientation of the camera. Thus, the extrinsic camera parameters change if the relative position or orientation between the specimen and camera change.

In contrast, the intrinsic camera parameters remain unchanged because they are only dependent on the camera system. The parameters ξ_x and ξ_y perform scaling from metric units to units of pixels. This paper uses millimetres as the metric units. The parameters c_x and c_y apply translation such that the origin of the sensor CS is at the top left of the image as shown in Figure 1. The parameter c_s converts from an orthogonal CS to a skewed sensor CS. Here, $c_s = 0$ since an orthogonal sensor CS is assumed. The parameter α is an arbitrary scaling variable of the homogeneous coordinates which is factored out. For more information on the pinhole camera model refer to the work of Zhang [41] and Heikkila et al. [42].

2.1.3. Radial Distortion Model

According to Tsai [43] and Wei et al. [44], the difference between the ideal and actual image can be well accounted for by using only a radial distortion model. Radial distortion is caused by the lens system having different magnification levels depending on where the light ray passes through the lenses. The image experiences either an increase (pincushion distortion) or decrease (barrel distortion) in magnification with increasing distance from the optical axis. The radial distortion model requires that $\underline{\hat{\boldsymbol{x}}}_s$ be converted to normalised ideal image coordinates, $\hat{\boldsymbol{x}}_n = \begin{bmatrix} \hat{x}_n & \hat{y}_n \end{bmatrix}^T$, using the inverse of the intrinsic parameter matrix as

$$\hat{\boldsymbol{x}}_n = \begin{bmatrix} \hat{x}_n \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}(\boldsymbol{K}^{-1}\underline{\hat{\boldsymbol{x}}}_s). \tag{2}$$

This equation includes a matrix to convert from homogeneous coordinates to Cartesian coordinates. $\hat{\boldsymbol{x}}_n$ is related to the normalised, distorted image coordinates, $\boldsymbol{x}_n = \begin{bmatrix} x_n & y_n \end{bmatrix}^T$, as [41]

$$\boldsymbol{x}_n = (1 + \kappa_1\hat{\boldsymbol{x}}_n^T\hat{\boldsymbol{x}}_n + \kappa_2(\hat{\boldsymbol{x}}_n^T\hat{\boldsymbol{x}}_n)^2)\hat{\boldsymbol{x}}_n, \tag{3}$$

where κ_1 and κ_2 are the unit-less radial distortion parameters that quantify the severity of the distortion. $\boldsymbol{x}_n$ is converted to distorted coordinates in the distorted sensor CS, $\boldsymbol{x} = \begin{bmatrix} x & y \end{bmatrix}^T$, as

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \boldsymbol{K} \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix}. \tag{4}$$

2.1.4. Calibration Process

Calibration determines the extrinsic, intrinsic and radial distortion parameters using images taken of a calibration plate. A calibration plate is an object with a flat surface having a high contrast regular pattern which contains distinctive, point-like features called calibration targets (CTs). It is used to define a set of 3D coordinates in the world CS and a corresponding set of distorted, 2D coordinates in the distorted sensor CS.

The 3D coordinates of these CTs in the world CS are predefined. In fact, they lie on the x-y plane of the world CS and define its position and orientation. The set of corresponding distorted, 2D coordinates in the sensor CS can be determined by locating the CTs in an image taken of the calibration plate. These two sets of 3D and 2D coordinates are used to solve for the parameters of the camera model, which describe the relationship between the two. This is done in two steps.

The first step determines initial estimates for the extrinsic and intrinsic camera parameters using the closed form solution method proposed by Zhang [41]. The initial estimate of the radial distortion parameters is set to zero.

The second step works with two sets of CTs in the distorted sensor CS: the true CTs, $\boldsymbol{x}^{true} = \begin{bmatrix} x^{true} & y^{true} \end{bmatrix}^T$, obtained directly from the calibration images and the calculated CTs, $\boldsymbol{x}^{calc} = \begin{bmatrix} x^{calc} & y^{calc} \end{bmatrix}^T$, obtained by transforming the known CTs of the world CS to the distorted sensor CS using the camera model and the current estimate of the calibration parameters. The difference between the true and calculated CTs is quantified as the total projection error, E_{proj}, given as

$$E_{proj} = \sum_{l=1}^{L} \sum_{m=1}^{M} \left(\left(x_{lm}^{calc} - x_{lm}^{true} \right)^2 + \left(y_{lm}^{calc} - y_{lm}^{true} \right)^2 \right). \tag{5}$$

There are L many calibration images and M many CTs per calibration image. The second step uses iterative non-linear least-squares optimisation to solve for the calibration parameters which minimise E_{proj}. Note that multiple calibration images are used in order to form an over-determined system of equations. This makes the calibration process less sensitive to noise inherent in the images. For more information on the calibration process refer to the work of Zhang [41] and Heikkila et al. [42].

The last process in calibration corrects $\boldsymbol{T}$ for the thickness of the calibration plate, ρ, such that the x-y plane of the world CS is coincident with the surface of the specimen under consideration. The corrected translation vector, $\boldsymbol{T}_{spec}$, that replaces $\boldsymbol{T}$ in Equation (1) is determined as

$$\boldsymbol{T}_{spec} = \boldsymbol{T} - \boldsymbol{R} \begin{bmatrix} 0 \\ 0 \\ \rho \end{bmatrix}, \tag{6}$$

where $\boldsymbol{T}$ and $\boldsymbol{R}$ are the translation vector and rotation matrix determined by the above calibration process.

2.2. Correlation

Correlation considers two images: a reference image, F, representing the specimen at time $t = 0$, and a deformed image, G, representing the specimen at time $t = 1$. F is broken up into subsets which are groups of neighbouring pixels. Conceptually, correlation attempts to determine how a reference subset (*f*) must displace and deform such that it matches a corresponding subset, the investigated subset (*g*) in *G*. In practice, however, *f* remains unchanged while its pixel centre positions (hereafter referred to as pixel positions) are displaced and deformed according to $\boldsymbol{W}$, a predefined SF, resulting in the query points of the investigated subset. The investigated subset is obtained by sampling the deformed image at these query points. To better understand this, some details of correlation need to be explained.

Correlation operates in the distorted sensor CS, as illustrated in Figure 2. *f*'s centre position, $\boldsymbol{x}^o = \begin{bmatrix} x^o & y^o \end{bmatrix}^T$, has been displaced by u and v in the x- and y-direction, respectively, to obtain *g*'s centre position, $\boldsymbol{x}^d = \begin{bmatrix} x^d & y^d \end{bmatrix}^T$. The ith pixel position of *f*, given by $\boldsymbol{x}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^T$, is based on $\boldsymbol{x}^o$ and the distance from $\boldsymbol{x}^o$ to $\boldsymbol{x}_i$, $\Delta\boldsymbol{x}_i = \begin{bmatrix} \Delta x_i & \Delta y_i \end{bmatrix}^T$, as

$$\boldsymbol{x}_i = \Delta\boldsymbol{x}_i + \boldsymbol{x}^o = \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} \Delta x_i \\ \Delta y_i \end{bmatrix} + \begin{bmatrix} x^o \\ y^o \end{bmatrix}. \quad (7)$$

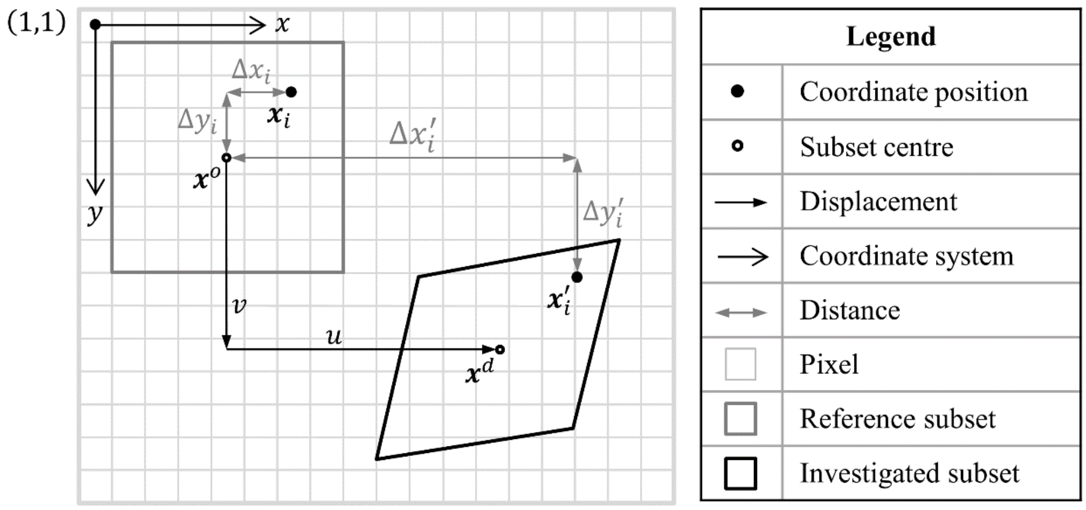


Figure 2. Schematic diagram illustrating how the pixel positions of the reference and investigated subsets are related to one another within the distorted sensor CS.

Similarly, the corresponding ith query point of *g*, $\boldsymbol{x}_i' = \begin{bmatrix} x_i' & y_i' \end{bmatrix}^T$, is based on $\boldsymbol{x}^o$ and the distance from $\boldsymbol{x}^o$ to $\boldsymbol{x}_i'$, $\Delta\boldsymbol{x}_i' = \begin{bmatrix} \Delta x_i' & \Delta y_i' \end{bmatrix}^T$, as

$$\boldsymbol{x}_i' = \Delta\boldsymbol{x}_i' + \boldsymbol{x}^o = \begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} \Delta x_i' \\ \Delta y_i' \end{bmatrix} + \begin{bmatrix} x^o \\ y^o \end{bmatrix}. \quad (8)$$

$\Delta\boldsymbol{x}_i'$ is defined relative to $\boldsymbol{x}^o$ because $\boldsymbol{x}^d$ is unknown prior to correlation. u and v are a special case of $\Delta x_i'$ and $\Delta y_i'$ for the pixel at the centre of the investigated subset. $\Delta\boldsymbol{x}_i'$ is determined using $\boldsymbol{W}$ which modifies $\Delta\boldsymbol{x}_i$ according to a given displacement and deformation quantified by the shape function parameters (SFPs), $\boldsymbol{P}$, as

$$\Delta\boldsymbol{x}_i' = \boldsymbol{W}(\Delta\boldsymbol{x}_i, \boldsymbol{P}). \quad (9)$$

Each pixel of the investigated subset, g_i, is populated by sampling the light intensity of the deformed image at $\boldsymbol{x}_i'$. However, images are discrete and so interpolation must be used to obtain these light intensities of *G* at non-integer locations. As such, *F* and *G* are treated as functions which return the light intensity at a location in the image. For *G* this involves interpolation. The pixels of *f* and *g* are populated by sampling these functions as

$$f_i = F(\boldsymbol{x}^o + \Delta\boldsymbol{x}_i) \text{and } g_i = G(\boldsymbol{x}^o + \boldsymbol{W}(\Delta\boldsymbol{x}_i, \boldsymbol{P})). \quad (10)$$

The similarity between $\boldsymbol{f}$ and $\boldsymbol{g}$ is quantified by the correlation criterion. Correlation aims to find the SFPs which define an investigated subset which closely matches the reference subset.

2.2.1. Correlation Criterion

The two most popular types are the ZNSSD and zero-mean normalised cross-correlation (ZNCC) criteria, which are robust against offset and scaling changes in light intensity. The ZNSSD criterion, which has a range of $\{C_{ZNSSD} \in \mathbb{R} | 0 \leq C_{ZNSSD} \leq 4\}$, where 0 indicates a perfect match, is calculated as

$$C_{ZNSSD} = \sum_{i=1}^{I} \left[\frac{f_i - \overline{f}}{\widetilde{f}} - \frac{g_i - \overline{g}}{\widetilde{g}} \right]^2, \tag{11}$$

where I is the number of pixels contained within a subset, $\overline{f} = \frac{\sum_i^I f_i}{I}$ and $\overline{g} = \frac{\sum_i^I g_i}{I}$ are the mean light intensity values, and $\widetilde{f} = \sqrt{\sum_{i=1}^{I} (f_i - \overline{f})^2}$ and $\widetilde{g} = \sqrt{\sum_{i=1}^{I} (g_i - \overline{g})^2}$ are the normalisation functions of subsets $\boldsymbol{f}$ and $\boldsymbol{g}$, respectively. Similarly, the ZNCC criterion, which has a range of $\{C_{ZNCC} \in \mathbb{R} | -1 \leq C_{ZNCC} \leq 1\}$, where 1 indicates a perfect match, is given as

$$C_{ZNCC} = \sum_{i=1}^{I} \frac{(f_i - \overline{f})(g_i - \overline{g})}{\widetilde{f}\widetilde{g}} \tag{12}$$

Pan et al. [45] proved that these two criteria are related as

$$C_{ZNCC} = 1 - \frac{C_{ZNSSD}}{2}. \tag{13}$$

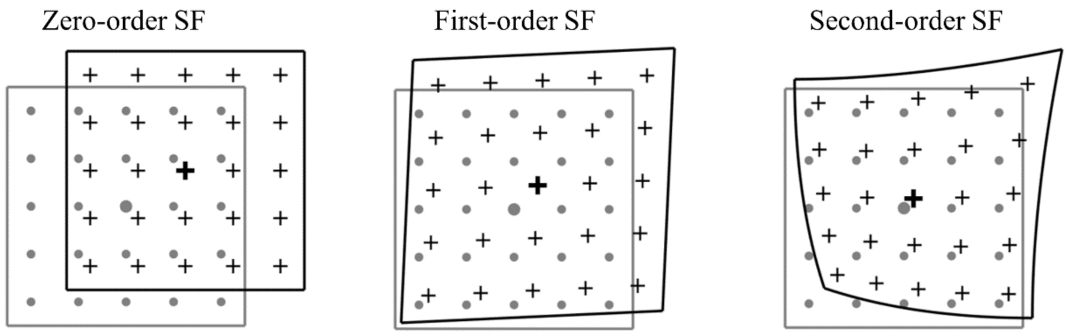
The more computationally efficient ZNSSD criterion is evaluated within ADIC2D; however, it is reported as the ZNCC coefficient, using Equation (13), because its range is more intuitive. For more information on correlation criteria refer to the work of Pan et al. [45].

2.2.2. Shape Function

The most common SFs are the zero ($\boldsymbol{W}^{SF0}$), first ($\boldsymbol{W}^{SF1}$) and second-order SFs ($\boldsymbol{W}^{SF2}$) expressed as [32]

$$\begin{gathered}
\boldsymbol{W}^{SF0}(\Delta \boldsymbol{x}_i, \boldsymbol{P}^{SF0}) = \begin{bmatrix} 1 & 0 & u \\ 0 & 1 & v \end{bmatrix} \begin{bmatrix} \Delta x_i \\ \Delta y_i \\ 1 \end{bmatrix}, \\
\boldsymbol{W}^{SF1}(\Delta \boldsymbol{x}_i, \boldsymbol{P}^{SF1}) = \begin{bmatrix} 1 + u_x & u_y & u \\ v_x & 1 + v_y & v \end{bmatrix} \begin{bmatrix} \Delta x_i \\ \Delta y_i \\ 1 \end{bmatrix} \\
\text{and } \boldsymbol{W}^{SF2}(\Delta \boldsymbol{x}_i, \boldsymbol{P}^{SF2}) = \begin{bmatrix} \frac{1}{2}u_{xx} & u_{xy} & \frac{1}{2}u_{yy} & 1 + u_x & u_y & u \\ \frac{1}{2}v_{xx} & v_{xy} & \frac{1}{2}v_{yy} & v_x & 1 + v_y & v \end{bmatrix} \begin{bmatrix} \Delta x_i^2 \\ \Delta x_i \Delta y_i \\ \Delta y_i^2 \\ \Delta x_i \\ \Delta y_i \\ 1 \end{bmatrix},
\end{gathered} \tag{14}$$

where u and v represent the displacement of $\boldsymbol{x}^o$ in the x- and y-directions respectively, and their derivatives (subscript x and y) define the deformation with respect to the reference subset. Specifically, u_x, u_{xx}, v_y and v_{yy} represent elongation while u_y, v_x, u_{yy}, v_{xx}, u_{xy} and v_{xy} represent shearing of the subset. Higher order SFs, containing higher order displacement derivatives, allow for more complex deformation as shown in Figure 3.



Legend	Subset outline	Pixel positions	Subset centre
Reference subset	—	•	●
Investigated subset	**—**	+	**+**

Figure 3. Schematic diagram illustrating the allowable deformation of a subset for various SF orders.

This enables higher order SFs to more reliably track subsets in complex displacement fields. The elements of $\boldsymbol{P}$, for each SF order, are stored as

$$\begin{aligned} \boldsymbol{P}^{SF0} &= \begin{bmatrix} u & v \end{bmatrix}^T, \\ \boldsymbol{P}^{SF1} &= \begin{bmatrix} u & u_x & u_y & v & v_x & v_y \end{bmatrix}^T \\ \text{and } \boldsymbol{P}^{SF2} &= \begin{bmatrix} u & u_x & u_y & u_{xx} & u_{xy} & u_{yy} & v & v_x & v_y & v_{xx} & v_{xy} & v_{yy} \end{bmatrix}^T. \end{aligned} \tag{15}$$

2.2.3. Interpolation

Interpolation determines the value at a query point (x'_i) in an image by fitting an equation to the surrounding light intensity data and evaluating the equation at x'_i. Polynomial interpolation and b-spline interpolation, shown in Figure 4 for the one-dimensional case, are the most popular types for DIC.

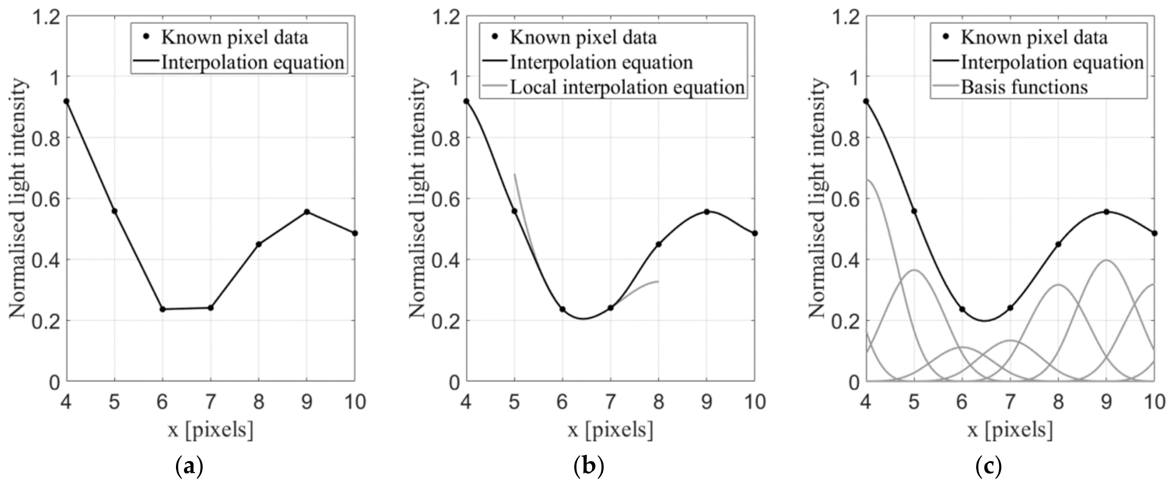


Figure 4. Graphical representation of the interpolation equations for: (**a**) linear polynomial; (**b**) cubic polynomial; and (**c**) cubic b-spline interpolation methods.

Polynomial interpolation fits a local polynomial equation of order n to a window of data of size $n + 1$ as shown in grey in Figure 4b for cubic polynomial interpolation. The resulting interpolation equation is a piecewise polynomial where only the central portion of each local polynomial equation is used. The interpolation equation is C^0 and C^1 continuous for linear and cubic polynomial interpolation, respectively. Refer to the work of Keys [46] for more information on cubic polynomial interpolation.

In contrast, b-spline interpolation builds up an interpolation equation from locally supported basis functions. More specifically, a basis function is defined at each data point and the coefficients of all these basis functions are determined simultaneously from the data. This is done such that the summation of the basis functions forms the interpolation equation as shown in Figure 4c. For cubic b-spline, the interpolation equation is C^2 continuous. Refer to the work of Hou et al. [47] for an in-depth discussion of bi-cubic b-spline interpolation.

The interpolation method should be as exact as possible in order for correlation to determine sub-pixel displacements reliably and efficiently because interpolation is the most time consuming part of correlation for iterative, sub-pixel DIC [48].

2.2.4. Gaussian Filtering

High order interpolation methods, such as bi-cubic b-spline interpolation, are sensitive to high frequency noise contained in the images [49]. A Gaussian low-pass filter is used to attenuate the high frequency noise of each image of the image set in order to reduce the bias of the displacement results caused by the interpolation method. Gaussian filtering convolves a 2D Gaussian point-spread function with the image. The Gaussian function consists of a window size, β (in pixels), and standard deviation, σ^g, to determine a weighted average light intensity at each pixel position in the filtered image from a window of pixels in the unfiltered image. The Gaussian point-spread function is scaled such that the sum of itself equals 1.

Although interpolation is only required for G, all the images of the image set (including F) need to be filtered such that the light intensity patterns of the subsets, considered by the correlation criterion, are directly comparable. Despite the fact that variance of the displacement results is independent of the interpolation method, it is dependent on the image detail which is reduced by smoothing [50]. Therefore β and σ^g should be chosen to reduce bias while not significantly increasing variance. For more information on Gaussian filtering refer to Pan's work [49].

2.2.5. Optimisation Method

The optimisation problem aims to minimise the correlation criterion (Equation (11)) by using the IC-GN method to iteratively solve for the optimal SFPs. An illustration of this process is shown in Figure 5. Substituting Equation (10) into Equation (11) results in an expression in terms of F and G being obtained. In addition, Equation (11) is modified to include an iterative improvement estimate, $\Delta \boldsymbol{P}$. Normally, iterative updating uses the forward additive implementation in which both $\Delta \boldsymbol{P}$ and $\boldsymbol{P}$ are applied to the investigated subset as $\boldsymbol{P} + \Delta \boldsymbol{P}$. However, for the inverse compositional implementation $\Delta \boldsymbol{P}$ is applied to the reference subset and the current estimate of $\boldsymbol{P}$ is applied to the investigated subset. Thus, the objective function is given as

$$C_{ObjFun} = \sum_{i=1}^{I} \left[\frac{F(\boldsymbol{x}^o + \boldsymbol{W}(\Delta \boldsymbol{x}_i, \Delta \boldsymbol{P})) - \overline{f}}{\widetilde{f}} - \frac{G(\boldsymbol{x}^o + \boldsymbol{W}(\Delta \boldsymbol{x}_i, \boldsymbol{P})) - \overline{g}}{\widetilde{g}} \right]^2 . \tag{16}$$

Taking the first-order Taylor series expansion of Equation (16) in terms of $\Delta \boldsymbol{P}$ gives

$$C_{ObjFun} = \sum_{i=1}^{I} \left[\frac{F(\boldsymbol{x}^o + \Delta \boldsymbol{x}_i) + \nabla \boldsymbol{f}_i \frac{\partial \boldsymbol{W}_i}{\partial \boldsymbol{P}} \Delta \boldsymbol{P} - \overline{f}}{\widetilde{f}} - \frac{G(\boldsymbol{x}^o + \boldsymbol{W}(\Delta \boldsymbol{x}_i, \boldsymbol{P})) - \overline{g}}{\widetilde{g}} \right]^2 , \tag{17}$$

where $\nabla f_i = \begin{bmatrix} \frac{\partial f_i}{\partial x} & \frac{\partial f_i}{\partial y} \end{bmatrix}$ is the light intensity gradient of f and $\frac{\partial \boldsymbol{W}_i}{\partial \boldsymbol{P}}$ is the Jacobian of the SF at each pixel position. For the zero, first and second-order SFs $\frac{\partial \boldsymbol{W}_i}{\partial \boldsymbol{P}}$ is given as [32,33]

$$\begin{gathered}\frac{\partial \boldsymbol{W}_i^{SF0}}{\partial \boldsymbol{P}^{SF0}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \\ \frac{\partial \boldsymbol{W}_i^{SF1}}{\partial \boldsymbol{P}^{SF1}} = \begin{bmatrix} 1 & \Delta x_i & \Delta y_i & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta x_i & \Delta y_i \end{bmatrix} \\ \text{and } \frac{\partial \boldsymbol{W}_i^{SF2}}{\partial \boldsymbol{P}^{SF2}} = \begin{bmatrix} 1 & \Delta x_i & \Delta y_i & \frac{\Delta x_i^2}{2} & \Delta x_i \Delta y_i & \frac{\Delta y_i^2}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta x_i & \Delta y_i & \frac{\Delta x_i^2}{2} & \Delta x_i \Delta y_i & \frac{\Delta y_i^2}{2} \end{bmatrix}.\end{gathered} \tag{18}$$

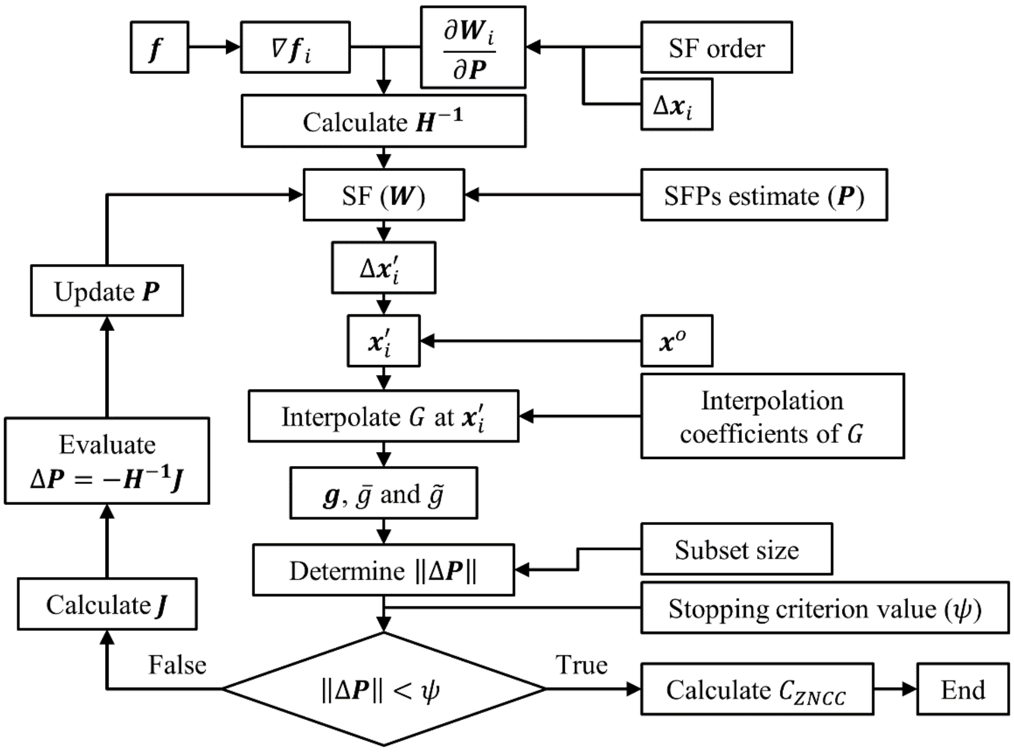


Figure 5. Flow diagram of ADIC2D's correlation process.

Setting Equation (17) to zero and taking the derivative with respect to $\Delta \boldsymbol{P}$ gives the first-order, least-squares solution. Rearranging to make $\Delta \boldsymbol{P}$ the subject of the equation yields

$$\Delta \boldsymbol{P} = -\boldsymbol{H}^{-1} \sum_{i=1}^{I} \left(\nabla f_i \frac{\partial \boldsymbol{W}_i}{\partial \boldsymbol{P}}\right)^T \left[f_i - \bar{f} - \frac{\tilde{f}}{\tilde{g}} (G(\boldsymbol{x}^o + \boldsymbol{W}(\Delta \boldsymbol{x}_i, \boldsymbol{P})) - \bar{g}) \right], \tag{19}$$

where $\boldsymbol{H}$ is the Hessian given by Equation (20) and the remaining terms, within the summation, of Equation (19) form the Jacobian, $\boldsymbol{J}$. $\boldsymbol{H}$ is independent of the SFPs and remains constant during iterations. Thus, Equation (20) can be pre-computed before iterations begin.

$$\boldsymbol{H} = \sum_{i=1}^{I} \left[\left(\nabla f_i \frac{\partial \boldsymbol{W}_i}{\partial \boldsymbol{P}}\right)^T \left(\nabla f_i \frac{\partial \boldsymbol{W}_i}{\partial \boldsymbol{P}}\right) \right]. \tag{20}$$

Note that since $\Delta \boldsymbol{P}$ is applied to the reference subset, each iteration solves for a set of SFPs which if applied to the reference subset would improve the correlation criterion. However, instead of applying $\Delta \boldsymbol{P}$ to the reference subset it is used to improve the estimate of the SFPs of the investigated subset.

More specifically, the updated SFPs of the investigated subset, $\boldsymbol{P}_{update}$, are obtained by composing the inverted iterative improvement, $\Delta\boldsymbol{P}$, with the current estimate, $\boldsymbol{P}$, as

$$\boldsymbol{P}_{update} = \boldsymbol{\omega}(\boldsymbol{P})\ \boldsymbol{\omega}(\Delta\boldsymbol{P})^{-1}, \tag{21}$$

where $\boldsymbol{\omega}$ is a function which populates a square matrix with the values of the SFPs as [36]

$$\begin{aligned}
\boldsymbol{\omega}^{SF0}(\boldsymbol{P}^{SF0}) &= \begin{bmatrix} 1 & 0 & u \\ 0 & 1 & v \\ 0 & 0 & 1 \end{bmatrix}, \\
\boldsymbol{\omega}^{SF1}(\boldsymbol{P}^{SF1}) &= \begin{bmatrix} 1+u_x & u_y & u \\ v_x & 1+v_y & v \\ 0 & 0 & 1 \end{bmatrix} \\
\text{and } \boldsymbol{\omega}^{SF2}(\boldsymbol{P}^{SF2}) &= \begin{bmatrix} 1+A_1 & A_2 & A_3 & A_4 & A_5 & A_6 \\ A_7 & 1+A_8 & A_9 & A_{10} & A_{11} & A_{12} \\ A_{13} & A_{14} & 1+A_{15} & A_{16} & A_{17} & A_{18} \\ \frac{1}{2}u_{xx} & u_{xy} & \frac{1}{2}u_{yy} & 1+u_x & u_y & u \\ \frac{1}{2}v_{xx} & v_{xy} & \frac{1}{2}v_{yy} & v_x & 1+v_y & v \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},
\end{aligned} \tag{22}$$

where A_1 through A_{18} are

$$\begin{aligned}
A_1 &= 2u_x + u_x^2 + uu_{xx}, & A_2 &= 2uu_{xy} + 2(1+u_x)u_y, \\
A_3 &= u_y^2 + uu_{yy}, & A_4 &= 2u(1+u_x), \\
A_5 &= 2uu_y, & A_6 &= u^2, \\
A_7 &= \tfrac{1}{2}(vu_{xx} + 2(1+u_x)v_x + uv_{xx}), & A_8 &= u_y v_x + u_x v_y + vu_{xy} + uv_{xy} + v_y + u_x, \\
A_9 &= \tfrac{1}{2}(vu_{yy} + 2(1+v_y)u_y + uv_{yy}), & A_{10} &= v + vu_x + uv_x, \\
A_{11} &= u + vu_y + uv_y, & A_{12} &= uv, \\
A_{13} &= v_x^2 + vv_{xx}, & A_{14} &= 2vv_{xy} + 2v_x(1+v_y), \\
A_{15} &= 2v_y + v_y^2 + vv_{yy}, & A_{16} &= 2vv_x, \\
A_{17} &= 2v(1+v_y) & \text{and } A_{18} &= v^2.
\end{aligned}$$

The optimisation method is computationally efficient because before iterations begin the following are computed: (i) $\boldsymbol{H}$ and its inverse; (ii) the interpolation coefficients of G; and (iii) the image gradients of F using the Prewitt gradient operator. Each iteration step involves evaluating $\boldsymbol{W}$ (Equation (14)) using the current estimate of $\boldsymbol{P}$ to obtain $\Delta\boldsymbol{x}_i'$, which is used by Equation (8) to compute $\boldsymbol{x}_i'$, interpolating G at $\boldsymbol{x}_i'$ in order to compute $\boldsymbol{g}$, $\overline{g}$ and $\widetilde{g}$ and finally computing $\Delta\boldsymbol{P}$ using Equation (19). For each iteration $\boldsymbol{P}$ is updated using Equation (21). Iterations continue until the stopping criterion deems that $\boldsymbol{P}$ is a solution. The correlation coefficient is then computed using Equation (11) substituted into Equation (13) and u and v are obtained from the SFPs.

2.2.6. Stopping Criterion

Iterations stop once the change in SFPs, $||\Delta\boldsymbol{P}||$, is below a specified threshold referred to as the stopping criterion value (ψ) [35]. The expressions for $||\Delta\boldsymbol{P}||$ for the SF orders are [36]

$$\begin{aligned}
||\Delta\boldsymbol{P}^{SF0}|| &= \left[\Delta u^2 + \Delta v^2\right]^{0.5}, \\
||\Delta\boldsymbol{P}^{SF1}|| &= \left[\Delta u^2 + (\Delta u_x\zeta)^2 + (\Delta u_y\zeta)^2 + \Delta v^2 + (\Delta v_x\zeta)^2 + (\Delta v_y\zeta)^2\right]^{0.5} \\
\text{and } ||\Delta\boldsymbol{P}^{SF2}|| &= \Big[\Delta u^2 + (\Delta u_x\zeta)^2 + (\Delta u_y\zeta)^2 + (\tfrac{1}{2}\Delta u_{xx}\zeta)^2 + (\tfrac{1}{2}\Delta u_{xy}\zeta)^2 + (\tfrac{1}{2}\Delta u_{yy}\zeta)^2 + \\
&\quad \Delta v^2 + (\Delta v_x\zeta)^2 + (\Delta v_y\zeta)^2 + (\tfrac{1}{2}\Delta v_{xx}\zeta)^2 + (\tfrac{1}{2}\Delta v_{xy}\zeta)^2 + (\tfrac{1}{2}\Delta v_{yy}\zeta)^2\Big]^{0.5},
\end{aligned} \tag{23}$$

where $\zeta = \frac{\sqrt[2]{I}-1}{2}$ is the furthest distance from $\boldsymbol{x}^o$.

2.3. Displacement Transformation

Displacement transformation maps u and v from the distorted sensor CS to the world CS. First, the position of the investigated subset, x^d, is determined as

$$\begin{bmatrix} x^d \\ y^d \end{bmatrix} = \begin{bmatrix} x^o \\ y^o \end{bmatrix} + \begin{bmatrix} u \\ v \end{bmatrix}. \tag{24}$$

An exact analytical solution for the inverse of Equation (3) does not exist because it requires determining the roots of a polynomial of degree greater than four [51]. As such distortion is removed from the reference and investigated subset positions using non-linear, least-squares optimisation.

The resulting undistorted sensor coordinates of the subset before, $\hat{x}^o = \begin{bmatrix} \hat{x}^o & \hat{y}^o \end{bmatrix}^T$, and after deformation, $\hat{x}^d = \begin{bmatrix} \hat{x}^d & \hat{y}^d \end{bmatrix}^T$, are transformed to the world CS using the inverse of the pinhole camera model as

$$\frac{1}{\alpha}\begin{bmatrix} \hat{x}_w \\ \hat{y}_w \\ 1 \end{bmatrix} = \left(\begin{bmatrix} \xi_x & c_s & c_x \\ 0 & \xi_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & T_1 \\ R_{21} & R_{22} & T_2 \\ R_{31} & R_{32} & T_3 \end{bmatrix} \right)^{-1} \begin{bmatrix} \hat{x}_s \\ \hat{y}_s \\ 1 \end{bmatrix}. \tag{25}$$

The corrected translation vector determined by Equation (6) is used in Equation (25). The resulting position of the reference, $\hat{x}^o_w = \begin{bmatrix} \hat{x}^o_w & \hat{y}^o_w \end{bmatrix}^T$, and investigated subsets, $\hat{x}^d_w = \begin{bmatrix} \hat{x}^d_w & \hat{y}^d_w \end{bmatrix}^T$, in the world CS are used to determine the metric displacement experienced by the subset, $\begin{bmatrix} \hat{u}_w & \hat{v}_w \end{bmatrix}^T$, as

$$\begin{bmatrix} \hat{u}_w \\ \hat{v}_w \end{bmatrix} = \begin{bmatrix} \hat{x}^d_w \\ \hat{y}^d_w \end{bmatrix} - \begin{bmatrix} \hat{x}^o_w \\ \hat{y}^o_w \end{bmatrix}. \tag{26}$$

2.4. Strain Computation

Strains are computed from the gradients of the displacements determined using Equation (26). A method of smoothing displacements before differentiation is recommended because these displacements contain noise which is amplified by differentiation. The method of point-wise least-squares proposed by Pan et al. [52] fits a planar surface to a window of displacement data using linear, least-squares optimisation with the subset of interest located at the centre of the window. The resulting equation for the planar surface is differentiated to determine the displacement gradients for the subset of interest. This is done for each subset and these displacement gradients are used to calculate the strains.

3. Framework Implementation

The ADIC2D framework, provided in Appendix A, is called from the command prompt as "`ProcData = ADIC2D(FileNames, Mask, GaussFilt, StepSize, SubSize, SubShape, SFOrder, RefStrat, StopCritVal, WorldCTs, ImgCTs, rho)`" requiring input variables as defined in Table 1 and providing an output variable as a structured array containing data for each analysed image d and subset q as detailed in Table 2.

3.1. `ADIC2D` Function

`ADIC2D` is the main function and is outlined in Table 3. Its purpose is to set up the DIC problem and call the appropriate subroutines. `ADIC2D` defines variables on a per image and subset basis to allow for complete flexibility in assigning `Xos`, `SubSize`, `SubShape` and `SFOrder`, i.e., on a per subset basis. Although `ADIC2D` is capable of this, it assigns the same `SubSize`, `SubShape` and `SFOrder` to each subset (in line 8 based on the inputs) since this is the most common use case. Output variables are pre-assigned in line 8 to allow for the collection of input data used and efficient storage of computed variables. Note that the SFPs are stored in a vector `P` which corresponds to the size of the second-order

SFP vector in Equation (15). Thus, the second-order SFPs of P, not used by the specified SF order, remain zero.

Table 1. Description of the required input variables for the ADIC2D framework.

Variable	Variable Description
`FileNames`	Cell array of character vectors containing the image file names of the image set d. All images need to be the same size.
`Mask`	Logical matrix, which is the same size as the images, indicating which pixels should not be analysed during correlation.
`GaussFilt`	Define the standard deviation and window size for the Gaussian filter in pixels as [`FiltSigma`, `FiltSize`] respectively where $\{\texttt{FiltSigma} \in \mathbb{R}^+ \mid \texttt{FiltSigma} > 0\}$ and $\{\texttt{FiltSize} \in \mathcal{N}\}$.
`StepSize`	Step size in pixels $\{\texttt{StepSize} \in \mathcal{N}\}$.
`SubSize`	Subset size in pixels $\{\texttt{SubSize} = 2k + 1 \mid k \in \mathcal{N}\}$.
`SubShape`	Subset shape {`SubShape` ∈ 'Square', 'Circle'}.
`SFOrder`	Dictates the SF order $\{\texttt{SFOrder} \in \mathcal{Z} \mid 0 \leq \texttt{SFOrder} \leq 2\}$.
`RefStrat`	Logical statement dictating reference image strategy (Section 3.2).
`StopCritVal`	Defines the stopping criterion value $\{\texttt{StopCritVal} \in \mathbb{R}^+ \mid \texttt{StopCritVal} > 0\}$.
`WorldCTs`	Location of CTs in the world CS defined according to MATLAB's estimateCameraParameters function.
`ImgCTs`	Location of CTs in the sensor CS defined according to MATLAB's estimateCameraParameters function.
`rho`	Calibration plate thickness in millimetres.

Table 2. Accessing the output variables for image d (contained in `ProcData(d)`) and subset number q.

Variable	Variable Description
`ImgName`	Image name.
`ImgSize(b)`	Image size (b = 1 for rows and b = 2 for columns).
`ImgFilt(b)`	Standard deviation (b = 1) and window size (b = 2) for the Gaussian filter respectively in pixels.
`SubSize(q)`	Subset size in pixels.
`SubShape(q)`	Subset shape.
`SFOrder(q)`	SF order.
`Xos(b,q)`	Reference subset position in the distorted sensor CS (b = 1 for x^o and b = 2 for y^o).
`Xow(b,q)`	Reference subset position in the world CS (b = 1 for $\hat{x}^o_w$ and b = 2 for $\hat{y}^o_w$).
`P(b,q)`	SFPs (b = 1 for u and b = 7 for v).
`C(q)`	ZNCC coefficient.
`Uw(b,q)`	Displacement in the world CS (b = 1 for $\hat{u}_w$ and b = 2 for $\hat{v}_w$).
`Iter(q)`	Number of iterations until stopping criterion is satisfied (maximum of 100 iterations).
`CamParams`	Calibration parameters.

Table 3. `ADIC2D` algorithm summary.

Line Numbers	Task Performed
Lines 2–4	Compute image names, number of images and size of the first image;
Lines 5–6	Create regularly spaced reference subset positions, `Xos`;
Line 7	Remove subsets containing invalid pixels which are defined by `Mask`;
Line 8	Pre-assign `ProcData` structure;
Line 9	Call subroutine `ImgCorr` to perform image correlation;
Line 10	Call subroutine `CSTrans` to perform transformation from the distorted sensor CS to the world CS;

`ADIC2D` calls the subroutine `ImgCorr` to perform the image correlation as presented above. `ImgCorr`'s input variables are `n` (the total number of images in the set), the pre-assigned variables in `ProcData`, `FileNames`, `RefStrat` and `StopCritVal`. The output variables are `P`, `C`, `Iter` and `StopVal` which are stored in `ProcData`. The computed SFPs are then passed to `CSTrans` to transform displacements to the world CS. `CSTrans`'s input variables are `n`, `ProcData`, `WorldCTs`, `ImgCTs` and `rho`. The output variables are `Xow`, `Uw` and MATLAB's `CamParams` (containing the intrinsic, extrinsic, and radial distortion parameters) which are stored in `ProcData`. Note that within the subroutines `ProcData` is shortened to `PD`.

The presented framework assumes a constant, regularly spaced `Xos` defined using `StepSize` and `SubSize`. Subsets which contain pixels that `Mask` indicates should not be analysed are removed.

3.2. Correlation Implementation

Correlation is performed using five subroutines: (i) `ImgCorr`, which performs the correlation on an image bases, i.e., between F and G; (ii) `SubCorr`, which performs the correlation on a subset basis; (iii) `SFExpressions`, which defines anonymous functions based on the SF order; (iv) `SubShapeExtract`, which determines input data for `SubCorr` based on the subset shape, size and position; and (v) `PCM`, which determines initial estimates for the displacement SFPs.

`SubCorr`'s input variables are the interpolation coefficients, f_i, ∇f_i, `SubSize`, `SFOrder`, `Xos`, Δx_i, initial estimates for $\boldsymbol{P}$ and `StopCritVal`. Note that throughout Section 3.2 variables with subscript i refer to the full set of this variable for a subset (i.e., ∇f_i refers to $\nabla f_i \ \forall \ i \in I$). `SubCorr`'s output variables are `P`, `C`, `Iter` and `StopVal`. `SFExpressions`'s input variable is `SFOrder` with outputs as anonymous functions to compute $\boldsymbol{W}$, $\nabla f_i \frac{\partial W_i}{\partial \boldsymbol{P}}$ and $||\boldsymbol{P}||$. Moreover, two functions are included to compute $\boldsymbol{\omega}$ (given in Equation (22)) and to extract the SFPs from $\boldsymbol{\omega}$.

The framework considers two subset shapes, square and circular, which are commonly employed in subset based DIC. For circular subsets `SubSize` defines the diameter of the subset. `SubShapeExtract` is used to determine f_i, ∇f_i and Δx_i for a subset based on the inputs `SubSize`, `SubShape`, `Xos`, F, ∇F and `SubExtract`. ∇F is the light intensity gradient of the entire reference image and `SubExtract` is an anonymous function, defined in line 2 of `ImgCorr`, which extracts a square subset from a matrix based on the position and size of the subset. `PCM` returns u and v based on inputs F, G, `SubSize`, `Xos` (passed as two vectors as required by `arrayfun`) and `SubExtract`.

Furthermore, two reference strategies are considered, namely, an absolute and an incremental strategy. The absolute strategy defines the first image as F (i.e., `FileNames(1)`), whereas the incremental strategy defines the previous image as F (`FileNames(d-1)`). The incremental strategy handles large deformations between images more reliably; however, if total displacements are required, it suffers from accumulative errors. The variable `RefStrat` is set to 0 or 1 for the absolute or incremental strategy respectively. Alternate reference strategies may be set by modifying line 8 in `ImgCorr`.

Moreover, ADIC2D considers the zero, first and second-order SFs, as outlined in Section 2.2.2. Set `SFOrder` to 0, 1 or 2 for the zero, first and second-order SFs, respectively.

3.2.1. `ImgCorr` Function

`ImgCorr` uses two nested for-loops as summarised in Table 4. The outer loop cycles through the image set, whereas the inner loop cycles through the subsets. `ImgCorr` reads the appropriate image pairs F and G from the image set, depending on the chosen reference strategy, and filters both using MATLAB's `imgaussfilt` function. Alternate image filters can be employed by modifying line 5 and 9. Bi-cubic b-spline interpolation coefficients are computed using MATLAB's `griddedInterpolant` function. Alternate interpolation methods can be set by either modifying line 6 by replacing 'spline' with 'linear' or 'cubic', or replacing it with an alternate interpolation algorithm, such as MATLAB's `spapi` function for higher order spline interpolation. `griddedInterpolant` was used for computational efficiency.

For an incremental strategy, `Xos` is displaced using the displacement SFPs from the previous correlation run, to track the same light intensity patterns within the reference subsets. These displacements SFPs are rounded, as suggested by Zhou et al. [53], such that the pixel positions of the reference subset have integer values and avoid the need for interpolating the reference subset.

Table 4. `ImgCorr` algorithm summary.

Line Numbers	Task Performed
Line 2	Define `SubExtract` function to extract square subset data;
Line 3	**for** image number `d` = 2 to `d = n`, **do**
Line 4	Define G
Line 5	Perform Gaussian filtering on G using MATLAB's `imgaussfilt` function;
Line 6	Compute interpolation coefficients using MATLAB's `griddedInterpolant` function;
Line 7	**if** first image of correlation run **or** `Refstrat` is incremental, **do**
Line 8	Define F;
Line 9	Perform Gaussian filtering on F using MATLAB's `imgaussfilt` function;
Line 10	Compute gradients for F (compute ∇F);
Lines 11–12	Displace `Xos` with previous image correlation run displacements (incremental strategy);
Line 13	Call subroutine `PCM` to compute initial estimates of displacement SFPs;
Line 14	**else**, **do**
Line 15	Set `P(d)` $\leftarrow$ `P(d - 1)`;
Line 16	**end if**
Line 17	Initialise temporary storage variables used to save correlation information during the inner loop;
Line 18	**for** subset number `q = 1` to number of subsets, **do**
Line 19	Call subroutine `SubShapeExtract`;
Line 20	Call subroutine `SubCorr`;
Line 21	**end for**
Line 22	Save correlation information to `PD` variable;
Line 23–24	Show results for image `d` correlation;
Line 25	**end for**

Correlation of each subset requires SFP initial estimates. For the first run, ADIC2D uses a Phase Correlation Method (PCM) to determine initial estimates. Subsequent correlation runs use the previous correlation run's SFPs as an initial estimate. However, PCM is used for every run in the incremental strategy, as it allows for better stability if large displacements are expected. PCM can be used between each run by replacing line 15 with line 13. Moreover, alternate initial estimate strategies can be implemented by changing line 13. The PCM algorithm is discussed in Section 3.2.5.

The inner loop correlates each subset by using `SubShapeExtract` to determine the data for a subset while `SubCorr` uses this data to perform correlation of the subset. The loop can be implemented using parallel processing to reduce computation time by changing line 18 to a parfor-loop. However, during a parfor-loop the outputs of `SubCorr` cannot be saved directly to a structure variable. It is for this reason that they are saved to the temporary storage variables (initiated in line 17) during the loop and assigned to `PD` thereafter.

3.2.2. `SubShapeExtract` Function

`SubShapeExtract` returns the data sets of f_i, ∇f_i and Δx_i for a subset based on its intended shape, size and position, as outlined in Table 5. Note that these output data sets are in the form of vertical vectors. Alternative subset shapes can be added to this function provided they produce the same output data sets.

For a square subset `SubExtract` is used to extract the appropriate elements from the input matrices (F and ∇F) which correspond to the pixels of the subset. Δx_i is determined in line 7 according to `SubSize`.

For circular subsets the same process is followed. This results in temporary data sets f_i, ∇f_i and Δx_i which correspond to a square subset of size equal to the diameter of the intended circular subset. A mask identifying which elements, of these data sets, fall within the radius of the intended circular subset is computed in line 13 using Δx_i.This mask is used to extract the appropriate elements from the temporary data sets of the square subset resulting in the appropriate data sets for the circular subset.

Table 5. `SubShapeExtract` algorithm summary.

Line Numbers	Task Performed
Line 2	**switch** `SubShape`;
Line 3	**case** `SubShape = 'Square'`, **do**
Line 4–6	Extract f_i and ∇f_i using `SubExtract`;
Line 7	Compute Δx_i using `SubSize`;
Line 8	**case** `SubShape = 'Circle'`, **do**
Line 9–11	Extract f_i and ∇f_i using `SubExtract`;
Line 12	Compute Δx_i using `SubSize`;
Line 13	Determine mask of elements that fall within the circular subset;
Line 14–16	Use mask to extract appropriate data for circular subset;
Line 17	**end switch**

3.2.3. `SubCorr` Function

`SubCorr` is at the heart of ADIC2D and performs the subset-based correlation, as summarised in Table 6. It follows the theoretical framework presented in Section 2.2.

Table 6. `SubCorr` algorithm summary.

Line Numbers	Task Performed
Line 2	Call `SFExpressions` to assign equations dependent on the SF order;
Line 3	Compute $\nabla f_i \frac{\partial W_i}{\partial P}$;
Line 4	Compute $\boldsymbol{H}^{-1}$, Equation (20);
Line 5	Compute normalisation values $\overline{f}$ and $\widetilde{f}$;
Line 6	`Initialise flag` $\leftarrow$ `0, iter` $\leftarrow$ `0` and $\Delta \boldsymbol{P} \leftarrow 1$;
Line 7	**while** `flag = 0`, **do**
Line 8	Compute $\Delta x_i'$ Equation (14), using estimates of $\boldsymbol{P}$
Line 9	Compute g using interpolation coefficients;
Line 10	Compute normalisation values $\overline{g}$ and $\widetilde{g}$;
Line 11	Compute $\|\Delta \boldsymbol{P}\|$using Equation (23);
Line 12	**if** $\|\Delta \boldsymbol{P}\| <$ `StopCritVal` **or** `iter` > 100, **do**
Line 13	Set `iter` $\leftarrow$ `1`;
Line 14	Compute `C`, Equation (11) substituted into Equation (13);
Line 15	**else**, **do**
Line 16	Compute $\boldsymbol{J}$, Summation expression of Equation (19);
Line 17	Compute $\Delta \boldsymbol{P}$, Equation (19);
Line 18	Update $\boldsymbol{P}$, Equation (21);
Line 19	**end if**
Line 20	Set `iter` $\leftarrow$ `iter + 1`
Line 21	**end while**

3.2.4. `SFExpressions` Function

`SFExpressions` returns five anonymous functions based on the SF order specified and is outlined in Table 7. `W`, defines Equation (14), `dFdWdP` defines $\nabla f_i \frac{\partial W_i}{\partial P}$, `SFPVec2Mat` defines Equation (22), `Mat2SFPVec` extracts $\boldsymbol{P}$ from `SFPVec2Mat` and `StopCrit` defines Equation (23). Additional SFs, such as higher order polynomials, can be added after line 20 provided they are consistent with the outputs of `SFExpressions`.

Table 7. `SFExpressions` algorithm summary.

Line Numbers	Task Performed
Line 2	**switch** `SFOrder`
Line 3–8	**case** `SFOrder = 0`, **do** assign functions for zero-order SF;
Line 9–14	**case** `SFOrder = 1`, **do** assign functions for first-order SF;
Line 15–20	**case** `SFOrder = 2`, **do** assign functions for second-order SF;
Line 21	**end switch**

3.2.5. `PCM` Function

PCM performs correlation using the zero-order SF in the frequency domain to obtain initial displacement estimates. The algorithm is summarised in Table 8. `PCM` is efficient; however, it is limited to integer pixel displacements and can only use square subsets. Moreover, `PCM` is only capable of determining a reliable initial estimate if the displacement is less than half of `SubSize`. For more information on PCM, refer to the work of Foroosh et al. [54].

Table 8. `PCM` algorithm summary.

Line Numbers	Task Performed
Line 2	Compute normalised cross-power spectrum in the frequency domain;
Line 3	Convert back to spatial domain;
Line 4	Find index of the maximum correlation coefficient;
Line 5	Compute index vector which relates indices of the correlation coefficient matrix to the displacements they correspond to;
Line 6–7	Obtain displacements using index of the maximum correlation coefficient;

3.3. *`CSTrans` Function*

`CSTrans` performs CS and displacement transformations from the distorted sensor CS to the world CS as outlined in Table 9. `CSTrans` uses MATLAB's image calibration toolbox to determine calibration parameters according to Section 2.1 which are used to perform the transformations detailed in Section 2.3. Note that the extrinsic calibration parameters, extracted in line 8, are based on the final set of CTs in the sensor CS (`ImgCTs(:,:,end)`). Alternate calibration algorithms may be implemented by replacing lines 13 and 14.

Table 9. `CSTrans` algorithm summary.

Line Numbers	Task Performed
Line 2	**if** calibration targets are given, **do**
Line 3	Compute calibration parameters using MATLAB's `estimateCameraParameters` function;
Line 4	**else**, **do**
Line 5	Set unit calibration parameters and pass to MATLAB's `cameraParameters` function;
Line 6	Assign CTs in the distorted sensor and world CSs using `Xos`;
Line 7	**end if**
Line 8	Extract appropriate extrinsic parameters;
Line 9	Compute $\boldsymbol{T}_{spec}$, Equation (6);
Line 10	**for** image number d = 1 to d = n, **do**
Lines 11–12	Compute $\boldsymbol{x}_d$, Equation (24);
Line 13–14	Compute $\boldsymbol{x}_w^o$ and $\boldsymbol{x}_w^d$ using MATLAB's `undistortPoints` and `pointsToWorld` functions;
Line 15	Compute $\hat{u}_w$ and $\hat{v}_w$, Equation (26);
Line 16	Save calibration parameters;
Line 17	**end for**

4. Validation

ADIC2D was validated using the 2D DIC Challenge image sets that were created using TexGen [55] or Fourier methods [56] as documented by Reu et al. [39]. Homoscedastic Gaussian noise was applied to each image set to simulate camera noise. As stated by Reu et al. [39], "image noise is specified as one standard deviation of the grey level applied independently to each pixel". The respective noise levels are listed in Table 10. Samples 1–3 contain rigid body translations to assess the performance of the ADIC2D framework in the "ultimate error regime" [57]. This type of analysis aims to highlight the errors caused by contrast and noise, in the absence of complex displacement fields, interacting with the numerical processes of correlation [39,58]. Sample 14 contains a sinusoidal displacement field with increasing frequency. This type of analysis aims to highlight the compromise between noise suppression and spatial resolution (SR) [39].

Table 10. Details for the samples of the DIC Challenge [39].

Name	Method	Noise	Contrast	Images	Displacement field
Sample 1	TexGen	1.5	Varying	21	Shift of 0.05 pixels in both directions per image
Sample 2	TexGen	8	0–50	21	Shift of 0.05 pixels in both directions per image
Sample 3	Fourier	1.5	0–200	12	Shift of 0.1 pixels in both directions per image
Sample 14	Fourier	5	0–200	4	Sinusoid in the x-direction of increasing frequency (amplitude 0.1 pixels)

CS transformations were not performed during the validation process, by setting `WorldCTs = 0`, `ImageCTs = 0` and `rho = 0`. A stopping criterion of `StopCritVal = 10`$^{-4}$, limited to 100 iterations per subset (line 12 in `SubCorr`), was used. The Gaussian image filter was set to `FiltSize = 5` as this offers the best compromise between reducing bias and avoiding increasing variance [49]. `FiltSigma` is specified on a per sample basis.

4.1. Quantifying Error

Bias, variance, root-mean square error (RMSE) and SR were used to quantify errors. Bias refers to the mean of the absolute error (MAE_u, MAE_v) between the correlated and true values, while variance refers to the standard deviation of the absolute error (σ_u, σ_v). These are computed as

$$MAE_u = \frac{\sum_{q=1}^{Q}\left|u_q^{calc} - u_q^{true}\right|}{Q}, \quad MAE_v = \frac{\sum_{q=1}^{Q}\left|v_q^{calc} - v_q^{true}\right|}{Q}, \tag{27}$$

$$\sigma_u = \sqrt{\frac{\sum_{q=1}^{Q}\left(\left|u_q^{calc} - u_q^{true}\right| - MAE_u\right)^2}{Q-1}} \quad \text{and } \sigma_v = \sqrt{\frac{\sum_{q=1}^{Q}\left(\left|v_q^{calc} - v_q^{true}\right| - MAE_v\right)^2}{Q-1}} \tag{28}$$

where u_q^{calc} and v_q^{calc} are the correlated, u_q^{true} and v_q^{true} the true displacements in the x- and y-direction respectively and Q is total number of subsets. Bornert et al. [57] introduced a RMSE which summarises the full-field displacement errors as a single number calculated as

$$RMSE_u = \sqrt{\frac{\sum_{q=1}^{Q}\left(u_q^{calc} - u_q^{true}\right)^2}{Q}}, \quad \text{and } RMSE_v = \sqrt{\frac{\sum_{q=1}^{Q}\left(v_q^{calc} - v_q^{true}\right)^2}{Q}}. \tag{29}$$

Strain bias, variance and RMSE are calculated in the same way. SR is defined as the highest frequency of a sinusoidal displacement field at which the code is capable of capturing the peak displacements and strains within 95% and 90% of the true values, respectively [39]. SR is reported as the period such that lower values indicate better performance across all error metrics.

4.2. Samples 1–3

Samples 1–3 were correlated using ADIC2D, Justin Blaber's Ncorr (version 1.2) and LaVision's DaVis (version 8.4). Ncorr was used as it is well-established [59,60] and its correlation process is similar in theory to ADIC2D with the exception that it uses bi-quintic b-spline interpolation and the reliability-guided displacement tracking (RGDT) strategy proposed by Pan [61]. DaVis uses bi-sextic b-spline interpolation and was included to compare ADIC2D to a commercial software package.

ADIC2D was used by setting `SubShape = 'Circle'`, `StepSize = 5` pixels, `SFOrder = 1` and `SubSize` to 21, 41 and 81 pixels. A `FiltSigma` of 0.4, 0.8 and 0.6 was used for Samples 1, 2 and 3 respectively. Subsets of size 21, 41 and 81 pixels had 5000, 4500 and 3650 subsets per image respectively.

The following procedure is used to determine the error metrics for each sample on a per algorithm and per subset basis: (i) the displacement errors in the x- and y-direction were computed for each subset; (ii) these displacement errors were used to determine the error metrics in each direction for all subsets throughout the image set; and (iii) Pythagoras' theorem was used to determine the magnitude of each error metric, in pixels (pix), which is reported in Tables 11–13 for Samples 1–3, respectively.

Table 11. Sample 1 error analysis reported at $\times 10^{-3}$ pix.

Code	**Subset Size 21**			**Subset Size 41**			**Subset Size 81**		
	Bias	**Variance**	**RMSE**	**Bias**	**Variance**	**RMSE**	**Bias**	**Variance**	**RMSE**
ADIC2D	2.32	2.59	3.48	1.08	0.88	1.4	0.634	0.481	0.796
Ncorr	2.31	2.63	3.5	0.995	0.822	1.29	0.504	0.402	0.645
DaVis	1.34	1.05	1.7	0.652	0.497	0.819	0.317	0.246	0.401

Table 12. Sample 2 error analysis reported at $\times 10^{-1}$ pix.

Code	**Subset Size 21**			**Subset Size 41**			**Subset Size 81**		
	Bias	**Variance**	**RMSE**	**Bias**	**Variance**	**RMSE**	**Bias**	**Variance**	**RMSE**
ADIC2D	3.49	4.24	5.5	1.16	0.917	1.48	0.519	0.391	0.65
Ncorr	5.59	5.96	8.17	2.05	1.71	2.67	0.956	0.759	1.22
DaVis	1.84	1.57	2.42	0.781	0.599	0.985	0.377	0.281	0.47

Table 13. Sample 3 error analysis reported at $\times 10^{-3}$ pix.

Code	**Subset Size 21**			**Subset Size 41**			**Subset Size 81**		
	Bias	**Variance**	**RMSE**	**Bias**	**Variance**	**RMSE**	**Bias**	**Variance**	**RMSE**
ADIC2D	9.75	8.17	12.7	3.77	2.9	4.76	1.87	1.4	2.34
Ncorr	9.18	7.58	11.9	3.97	3.05	5.01	2.36	1.7	2.91
DaVis	5.16	4.02	6.54	2.47	1.9	3.12	1.29	1	1.64

Sample 1 reflects robustness to contrast changes, Sample 2 reflects robustness to higher noise content with a limited contrast range and Sample 3 reflects effects due to the interpolation method used.

ADIC2D's performance for Sample 1 is similar to that of Ncorr (up to 26% higher error). It is noted that ADIC2D does not use the RGDT strategy and therefore is somewhat more susceptible to contrast changes compared to Ncorr.

The improved performance of ADIC2D over Ncorr for Sample 2 (up to 48% improvement) is reasoned to be due to the SF order used to obtain an initial estimate of the SFPs. ADIC2D uses the zero-order SF resulting in reliable estimates of the SFPs. In contrast, Ncorr uses an overmatched first-order SF which causes estimates to take on spurious values, due to noise, causing correlation to proceed along an unfavourable solution path. Moreover, DaVis removes displacement results with poor correlation coefficients thereby not providing a true reflection of the overall error for Sample 2.

Sample 3 highlights the effect of lower order b-spline interpolation for smaller subsets. ADIC2D uses lower order b-spline interpolation, in comparison to Ncorr and DaVis, resulting in less accurate matching for smaller subsets (ADIC2D for `SubSize = 21` has a 7% higher RMSE relative to Ncorr).

4.3. Sample 14

ADIC2D was used for Sample 14 by setting `SubShape = 'Square'`, `StepSize` = 5, pixels, `SFOrder` = 1, `FiltSigma` = 0.4 and `SubSize` to 25, 31, 51 and 71 pixels. A window of 9 measurement points was used to compute strain data. Table 14 shows the displacement and strain results in the x-direction, for the last image of the set, that were analysed using the MATLAB code provided by the DIC Challenge [39]. Codes A and G published in [39], which exhibit the best noise suppression (variance) and SR, respectively, are included for comparison. Subsets of size 25, 31, 51 and 71 pixels had 43,700, 43,700, 42,600 and 40,600 subsets per image.

Table 14. Sample 14 error analysis in the x-direction for last image of the image set [39].

Code/Subset Size	Displacement				Strain			
	RMSE (pix)	Variance (pix)	Max Bias (pix)	SR (pix)	RMSE (pix/pix)	Variance (pix/pix)	Max Bias (pix/pix)	SR (pix)
Code G	0.010	0.010	0.012	100	453	429	923	74
ADIC2D 25	0.018	0.017	0.015	1629	598	406	1488	173
ADIC2D 31	0.014	0.013	0.017	160	600	335	1674	182
ADIC2D 51	0.014	0.007	0.033	257	839	193	2720	233
ADIC2D 71	0.022	0.005	0.059	354	1255	125	4412	294
Code A	0.022	0.005	0.056	716	1131	172	3399	410

ADIC2D is capable of dealing with high frequency displacement fields. For a subset size of 71 pixels ADIC2D performs similarly to code A (within 0.1% difference) with the exception of an improved SR (51%) and higher maximum bias (5%). As the subset size decreases so does the RMSE, bias and SR while variance increases. Figure 6 illustrates this increase in noise suppression with increase in subset size. For `SubSize` = 25 pixels, the error metrics increase (except strain SR as illustrated in Figure 7b), indicating a limitation of ADIC2D with regards to noise suppression and SR for smaller subset sizes (as shown in Figure 7a). The strain SR does not increase because strain experiences more spatial filtering than displacement for the reasons outlined in the DIC Challenge paper [39]. Although ADIC2D cannot achieve results similar to code G, the results in Table 14 indicate that the noise suppression and SR are within the range of established DIC codes evaluated in the DIC Challenge paper [39].

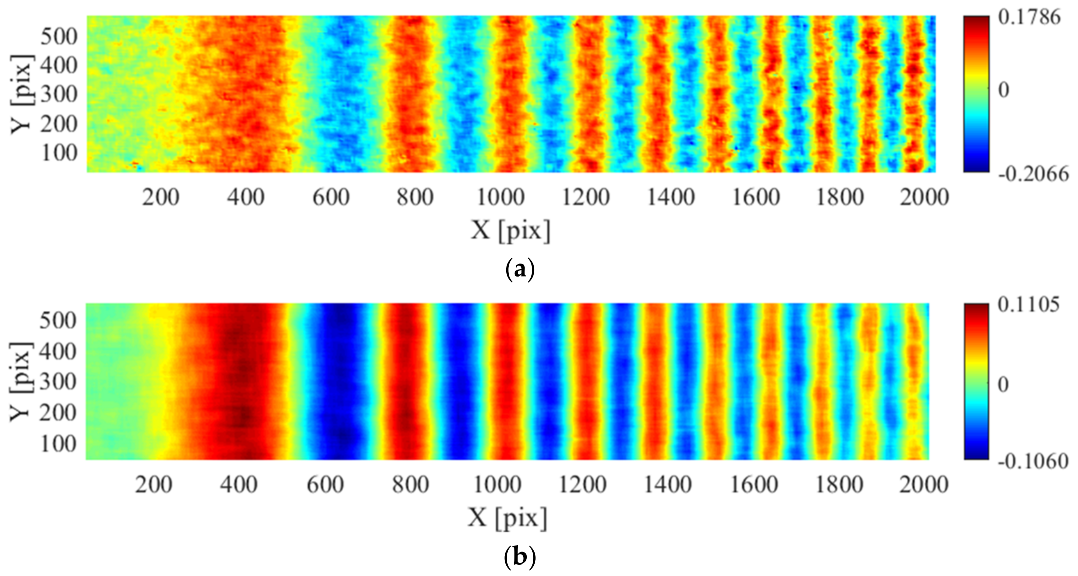


Figure 6. Comparison of the x-displacement for Sample 14 for a subset size of: (**a**) 25 pixels; and (**b**) 51 pixels.

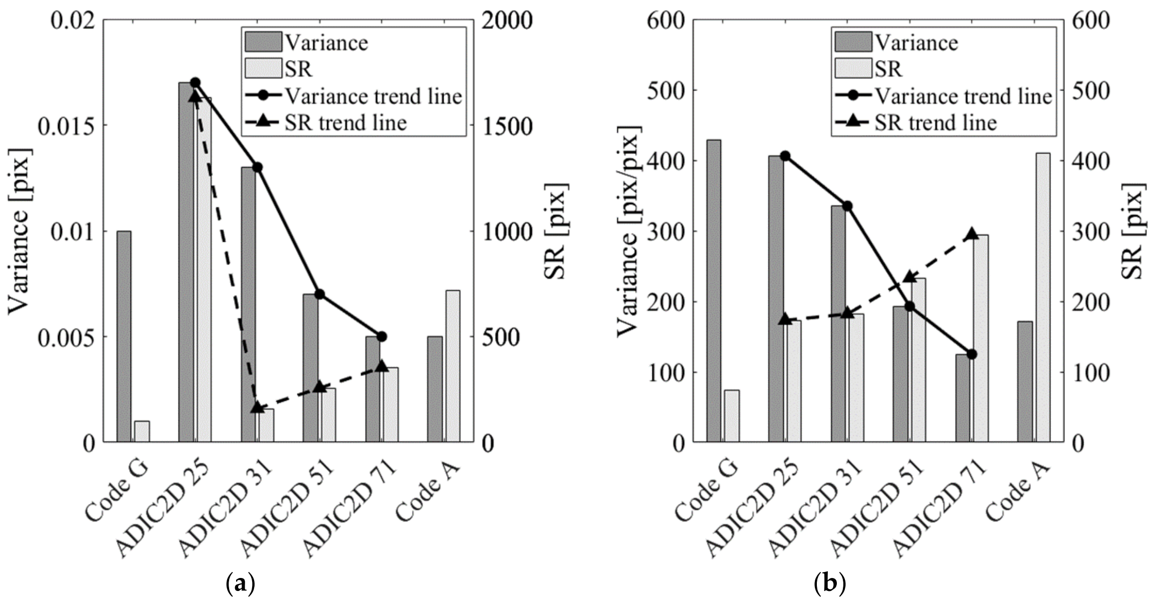


Figure 7. Comparison of the noise suppression (variance) and SR for various subset sizes in terms of: (**a**) displacement; and (**b**) strain results.

5. Discussion

The code was designed with modularity in mind. Firstly, it is modular in that each main task is performed by a separate subroutine such that the reader can progressively build up their understanding of the overall code by considering individual subroutines. This is particularly evident for the correlation subroutines which separate correlation such that the logistics of preparing data for correlation (`ImgCorr`), the core correlation operations (`SubCorr`), the effect of different SF orders on correlation (`SFExpressions`), how data sets are prepared for different subset shapes (`SubShapeExtract`) and determining initial estimates of the SFPs (`PCM`) can be considered separately.

Secondly, the code allows for changing of the SF order, subset shape, interpolation method and Gaussian filtering parameters. Although the effect of these on the displacement and strain results is well documented [31,49,62], this code allows the reader to easily investigate the effect of these in a practical manner.

The effect of the subset shape is subtle. The displacement determined at the centre of a subset is essentially the average of the displacement experienced by the light intensity pattern contained within the subset. However, the farther a pixel is from the subset centre, the less representative its displacement is of the displacement occurring at the subset centre. As such, circular subsets have become favoured since the pixels of their pixels are evenly distributed around the subset centre in a radially symmetric manner. However, since the trade-off is not significant and square subsets are simpler from a mathematical and programming viewpoint, many DIC algorithms still use square subsets.

Thirdly, the code is modular in that it allows the subset size, subset shape and SF order to be assigned on a per subset and per image basis. Traditionally, DIC makes use of a single subset size, subset shape and SF order for all subsets across all images. However, there has been a growing interest in the field of DIC to create algorithms which adaptively assign these parameters such that they are the most appropriate for the displacement and speckle pattern that the subset is attempting to track resulting in more reliable displacements being computed. The modularity of ADIC2D means it is straightforward to couple it with such an adaptive strategy.

In order to keep the code simple two aspects were neglected that would have otherwise made the correlation aspect of ADIC2D consistent with the current state-of-the-art as identified by Pan [2]. Firstly, ADIC2D makes use of bi-cubic b-spline interpolation, as opposed to the recommended bi-quintic b-spline interpolation. As stated in the work of Bornert et al. [57] the errors in the "ultimate error regime" are reduced by increasing the degree of the interpolation method particularly for smaller subsets. This is reflected in Table 13, which shows that although the error metrics of ADIC2D are better than that of Ncorr for larger subsets, the opposite is true for the subset size of 21 pixels.

Secondly, ADIC2D does not use the RGDT strategy. While ADIC2D uses the optimal SFPs of a subset for the previous image pair as an initial estimate of the SFPs for the current image pair, RGDT only does this for the subset with the best correlation coefficient for the previous image pair. It then uses the SFPs of this subset, for the current image pair, as initial estimates to correlate its neighbouring subsets. It then repeatedly identifies the subset with the best correlation coefficient, which has neighbouring subsets which have not yet been correlated, and uses its SFPs to correlate its neighbouring subsets. This is repeated until all the subsets have been correlated for the current image pair.

Thus, ADIC2D is susceptible to propagating spurious SFPs of a subset through the image set which the RGDT strategy would have avoided. The effect of this is reflected in the results of Table 11 which shows how ADIC2D struggles to perform as consistently as Ncorr in the presence of contrast changes in the image set.

Despite this, ADIC2D performs on par with established DIC algorithms. More specifically, (i) it is capable of dealing with contrast changes as shown in Table 11; (ii) it handles high levels of noise within the images sufficiently well as reflected in the results of Table 12; (iii) although displacement results of smaller subsets suffer due to its lower order bi-cubic b-spline interpolation, its interpolation method is sufficient achieving results similar to Ncorr as show in Table 13; and (iv) it has noise suppression and spatial resolution characteristics that fall within the range of those reported for established DIC algorithms as shown in Figure 7.

Thus, ADIC2D can be considered sufficiently reliable for use in the field of experimental solid mechanics. However, ADIC2D is not limited to this field since its modularity means it can be easily adapted for various applications and specific use cases. Furthermore, validation of ADIC2D coupled with its modularity not only makes it attractive as a learning resource, but also as a starting point to develop the capabilities of DIC.

6. Conclusions

This paper presents the theory of a 2D, subset based DIC framework (ADIC2D) that is predominantly consistent with current state-of-the-art techniques, and illustrates its numerical implementation in 117 lines of MATLAB code. ADIC2D allows for complete flexibility in assigning correlation attributes on a per image and per subset basis. ADIC2D includes Gaussian image filtering parameters, square or circular subset shape selection, zero, first and second-order SFs, reference image strategy selection, interpolation method flexibility, image calibration using MALAB's image calibration toolbox and displacement transformation. Moreover, the presented code is modular. Sections of the framework can readily be changed enabling the reader to gain a better understanding of DIC as well as to contribute to the development of new DIC algorithm capabilities. Validation of ADIC2D shows that it performs on par with established DIC algorithms.

Author Contributions: Conceptualization, T.B. and D.A.; Methodology, D.A. and T.B.; Software, D.A.; Validation, D.A.; Investigation, D.A.; Resources, T.B.; Writing—original draft preparation, D.A.; Writing—review and editing, T.B.; Visualization, D.A.; Supervision, T.B.; Funding acquisition, T.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Mechanical and Mechatronic Engineering department of the University of Stellenbosch.

Acknowledgments: The authors would like to thank D.Z. Turner for discussions and advice received during the compilation of this work as well as M.R. Molteno for advice received at the beginning of this work.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

α	Homogeneous scaling variable
β	Window size for the Gaussian filter
c_s	Skew of sensor coordinate system (intrinsic camera parameter)
$c_x,\ c_y$	Translation applied to sensor coordinate system (intrinsic camera parameter)
C_{ZNSSD}	Zero-mean normalised sum of squared difference correlation criterion
C_{ZNCC}	Zero-mean normalised cross correlation criterion
C_{ObjFun}	Objective function of correlation
E_{proj}	Total projection error
F	Reference image
$\boldsymbol{f}$	Reference subset
$\overline{f}$	Mean light intensity of reference subset
$\widetilde{f}$	Normalisation function of reference subset
∇F	Light intensity gradient of the reference image
$\nabla \boldsymbol{f}_i = \begin{bmatrix} \frac{\partial f_i}{\partial x} & \frac{\partial f_i}{\partial y} \end{bmatrix}$	Light intensity gradient of the reference subset
G	Deformed image
$\boldsymbol{g}$	Investigated subset
$\overline{g}$	Mean light intensity of investigated subset
$\widetilde{g}$	Normalisation function of investigated subset
$\boldsymbol{H}$	Hessian of the optimisation equation
I	Number of pixels per subset (counter is i)
$\boldsymbol{J}$	Jacobian of the optimisation equation
$\boldsymbol{K}$	Intrinsic camera parameters
κ_1, κ_2	Radial distortion parameters
L	Number of calibration images (counter is l)
M	Number of calibration targets per calibration image (counter is m)
$MAE_u,\ MAE_v$	Mean absolute error
$\boldsymbol{\omega}$	Function to populate a square matrix with the shape function parameters
ψ	Stopping criterion value
$\boldsymbol{P}$	Shape function parameters
$\boldsymbol{P}_{update}$	Updated shape function parameters
$\Delta\boldsymbol{P}$	Iterative improvement estimate of the shape function parameters
Q	Number of subsets per an image (counter is q)
$\boldsymbol{R}$	Rotation matrix
$RMSE_u,\ RMSE_v$	Root mean square error
ρ	Calibration plate thickness
σ^g	Standard deviation of the Gaussian function for Gaussian filtering
$\sigma_u,\ \sigma_v$	Variance (standard deviation of the absolute error)
t	Time
$\boldsymbol{T}$	Translation vector
$\boldsymbol{T}_{spec}$	Translation vector corrected for the thickness of the calibration plate
u	Displacement in the x-direction in the distorted sensor coordinate system
u^{true}	True displacement of subset in the x-direction in the distorted sensor coordinate system
u^{calc}	Calculated displacement of subset in the x-direction in the distorted sensor coordinate system
$\hat{u}_w$	Undistorted metric displacement in the x-direction in the world coordinate system
$u_x, u_y, u_{xx}, u_{xy}, u_{yy}$	Derivatives of the x-direction displacement
$\boldsymbol{V}$	Extrinsic camera parameters
v	Displacement in the y-direction in the distorted sensor coordinate system
v^{true}	True displacement of the subset in the y-direction in the distorted sensor coordinate system
v^{calc}	Calculated displacement of the subset in the y-direction in the distorted sensor coordinate system

$\hat{v}_w$	Undistorted metric displacement in the y-direction in the world coordinate system
$v_x, v_y, v_{xx}, v_{xy}, v_{yy}$	Derivatives of the y-direction displacement
$\boldsymbol{W}$	Shape function
$\frac{\partial \boldsymbol{W}_i}{\partial \boldsymbol{P}}$	Jacobian of the shape function in terms of the shape function parameters for pixel i
$\hat{\boldsymbol{x}}_w = \begin{bmatrix} \hat{x}_w & \hat{y}_w & \hat{z}_w \end{bmatrix}^T$	Ideal world coordinates
$\hat{\boldsymbol{x}}_s = \begin{bmatrix} \hat{x}_s & \hat{y}_s \end{bmatrix}^T$	Ideal sensor coordinates
$\hat{\boldsymbol{x}}_n = \begin{bmatrix} \hat{x}_n & \hat{y}_n \end{bmatrix}^T$	Normalized ideal image coordinates
$\boldsymbol{x}_n = \begin{bmatrix} x_n & y_n \end{bmatrix}^T$	Normalized distorted image coordinates
$\boldsymbol{x} = \begin{bmatrix} x & y \end{bmatrix}^T$	Distorted sensor coordinates
$\boldsymbol{x}^{true} = \begin{bmatrix} x^{true} & y^{true} \end{bmatrix}^T$	True location of the calibration targets in the distorted sensor coordinate system
$\boldsymbol{x}^{calc} = \begin{bmatrix} x^{calc} & y^{calc} \end{bmatrix}^T$	Location of the calibration targets in the distorted sensor coordinate system predicted by the camera model
$\boldsymbol{x}^o = \begin{bmatrix} x^o & y^o \end{bmatrix}^T$	Centre of reference subset in the distorted sensor coordinate system
$\hat{\boldsymbol{x}}^o = \begin{bmatrix} \hat{x}^o & \hat{y}^o \end{bmatrix}^T$	Centre of reference subset in the undistorted sensor coordinate system
$\boldsymbol{x}^d = \begin{bmatrix} x^d & y^d \end{bmatrix}^T$	Centre of investigated subset in the distorted sensor coordinate system
$\hat{\boldsymbol{x}}^d = \begin{bmatrix} \hat{x}^d & \hat{y}^d \end{bmatrix}^T$	Centre of investigated subset in the undistorted sensor coordinate system
$\boldsymbol{x}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^T$	ith pixel position of the reference subset in the distorted sensor coordinate system
$\boldsymbol{x}'_i = \begin{bmatrix} x'_i & y'_i \end{bmatrix}^T$	ith pixel position of the investigated subset in the distorted sensor coordinate system
$\Delta\boldsymbol{x}_i = \begin{bmatrix} \Delta x_i & \Delta y_i \end{bmatrix}^T$	Distance from the reference subset centre to ith pixel position of reference subset
$\Delta\boldsymbol{x}'_i = \begin{bmatrix} \Delta x'_i & \Delta y'_i \end{bmatrix}^T$	Distance from the reference subset centre to ith pixel position of investigated subset
$\hat{\boldsymbol{x}}^o_w = \begin{bmatrix} \hat{x}^o_w & \hat{y}^o_w \end{bmatrix}^T$	Undistorted reference subset position in the world coordinate system
$\hat{\boldsymbol{x}}^d_w = \begin{bmatrix} \hat{x}^d_w & \hat{y}^d_w \end{bmatrix}^T$	Undistorted investigated subset position in the world coordinate system
ξ_x, ξ_y	Scaling of metric units to pixels (intrinsic camera parameter)
ζ	Maximum distance of a pixel of the reference subset to the centre of the reference subset

Appendix A

The ADIC2D code can be accessed on GitHub at https://github.com/SUMatEng/ADIC2D.

```
function ProcData=ADIC2D(FileNames,Mask,GaussFilt,StepSize,SubSize,
SubShape,SFOrder,RefStrat,StopCritVal,WorldCTs,ImgCTs,rho)
[~,ImNames]=cellfun(@fileparts,FileNames,'Uni',0);
n=numel(FileNames);
[r,c]=size(im2double(imread(FileNames{1})));
[XosX,XosY]=meshgrid(((SubSize+1)/2+StepSize):StepSize:
(c-(SubSize+1)/2-1-StepSize),((SubSize+1)/2+StepSize):
StepSize:(r-(SubSize+1)/2-1-StepSize));
Xos=[XosX(:)'; XosY(:)'];
Xos=Xos(:,arrayfun(@(X,Y) min(min(Mask(Y-(SubSize-1)/2:Y+(SubSize-
1)/2,X-(SubSize-1)/2:X+(SubSize-1)/2))),Xos(1,:),Xos(2,:))==1);
ProcData=struct('ImgName',ImNames,'ImgSize',repmat({[r,c]},1,n),
'ImgFilt',repmat({GaussFilt},1,n),'SubSize',repmat({SubSize*ones([1,
size(Xos,2)])},1,n),'SubShape',repmat({repmat(SubShape,size(Xos,2),1)},
1,n),'SFOrder',repmat({repmat(SFOrder,size(Xos,2),1)},1,n),'Xos',repmat
({Xos},1,n),'P',repmat({zeros([12,size(Xos,2)])},1,n),'C',repmat({ones
([1,size(Xos,2)])},1,n),'StopVal',repmat({ones([1,size(Xos,2)])*StopCritVal},
1,n),'Iter',repmat({zeros([1,size(Xos,2)])},1,n));
ProcData=ImgCorr(n,ProcData,FileNames,RefStrat,StopCritVal); % Section 3.2.1
ProcData=CSTrans(n,ProcData,WorldCTs,ImgCTs,rho); % Section 3.3
```

```
function PD=ImgCorr(n,PD,ImNames,RefStrat,StopCritVal)
  SubExtract=@(Mat,Xos,SubSize) Mat(Xos(2)-(SubSize-
  1)/2:Xos(2)+(SubSize-1)/2,Xos(1)-(SubSize-1)/2:Xos(1)+(SubSize-1)/2);
  for d=2:n, tic; % outer loop
     G=im2double(imread(ImNames{d}));
     if all(PD(d).ImgFilt), G=imgaussfilt(G,PD(d).ImgFilt(1),
     'FilterSize',PD(d).ImgFilt(2)); end % Section 2.2.4
     InterpCoef=griddedInterpolant({1:1:size(G,1),1:1:size(G,2)}, G,'spline');
     % Section 2.2.3
     if any([RefStrat==1,d==2])
         F=im2double(imread(ImNames{d-1}));
         if all(PD(d).ImgFilt), F=imgaussfilt(F,PD(d).ImgFilt(1),
         'FilterSize',PD(d).ImgFilt(2)); end % Section 2.2.4
         [dFdx,dFdy]=imgradientxy(F,'prewitt');
         PD(d).Xos(1,:)=PD(d-1).Xos(1,:)+fix(PD(d-1).P(1,:));
         PD(d).Xos(2,:)=PD(d-1).Xos(2,:)+fix(PD(d-1).P(7,:));
         [PD(d).P(1,:),PD(d).P(7,:)]=arrayfun(@(XosX,XosY,SubSize)
         PCM(F,G,SubSize,XosX,XosY,SubExtract),PD(d).Xos(1,:),PD(d).Xos(
         2,:),PD(d).SubSize); % Section 3.2.5
     else
         PD(d).P=PD(d-1).P;
     end
     P=zeros(size(PD(d).P)); C=zeros(size(PD(d).C));
     Iter=zeros(size(PD(d).C)); StopVal=ones(size(PD(d).C));
     for q=1:size(PD(d).Xos,2) % inner loop (can be changed to parfor
     for parallel processing)
         [f,dfdx,dfdy,dX,dY]=SubShapeExtract(PD(d).SubSize(q),
         PD(d).SubShape(q,:),PD(d).Xos(:,q),F,dFdx,dFdy,SubExtract); % Section 3.2.2
         [P(:,q),C(q),Iter(q),StopVal(q)]=SubCorr(InterpCoef,f,dfdx,
         dfdy,PD(d).SubSize(q),PD(d).SFOrder(q),PD(d).Xos(:,q),dX,dY,PD(
         d).P(:,q),StopCritVal); % Section 3.2.3
     end
     PD(d).P=P; PD(d).C=C; PD(d).Iter=Iter; PD(d).StopVal=StopVal;
     if rem(d-2,10)==0, fprintf('Image/Total|  Time (s) | CC (min) | CC
     (mean) | Iter (max) \n'); end
     fprintf(' %4.d/%4.d |  %7.3f  | %.6f | %.6f  |  %3.0f
     \n',d,n,toc,min(PD(d).C),nanmean(PD(d).C),max(PD(d).Iter));
 end
```

```
function [f,dfdx,dfdy,dX,dY]=SubShapeExtract(SubSize,SubShape,Xos,F,
dFdx,dFdy,SubExtract)
switch SubShape
case 'Square'
    f(:)=reshape(SubExtract(F,Xos,SubSize),[SubSize*SubSize,1]);
    dfdx(:)=reshape(SubExtract(dFdx,Xos,SubSize), [SubSize*SubSize,1]);
    dfdy(:)=reshape(SubExtract(dFdy,Xos,SubSize), [SubSize*SubSize,1]);
    [dX,dY]=meshgrid(-(SubSize-1)/2:(SubSize-1)/2, -(SubSize-
    1)/2:(SubSize-1)/2); dX=dX(:); dY=dY(:);
 case 'Circle'
    f=SubExtract(F,Xos,SubSize);
    dfdx=SubExtract(dFdx,Xos,SubSize);
    dfdy=SubExtract(dFdy,Xos,SubSize);
    [dX,dY]=meshgrid(-(SubSize-1)/2:(SubSize-1)/2,-(SubSize-1)/2:(SubSize-1)/2);
    mask_keep=sqrt(abs(dX).^2+abs(dY).^2)<=(SubSize/2-0.5);
    f=f(mask_keep);
    dfdx=dfdx(mask_keep); dfdy=dfdy(mask_keep);
    dX=dX(mask_keep); dY=dY(mask_keep);
 end
```

```
function [P,C,iter,StopVal]=SubCorr(InterpCoef,f,dfdx,dfdy,SubSize,
SFOrder,Xos,dX,dY,P,StopCritVal)
[W,dFdWdP,SFPVec2Mat,Mat2SFPVec,StopCrit]=SFExpressions(SFOrder); % Section 3.2.4
dfdWdP=dFdWdP(dX(:),dY(:),dfdx(:),dfdy(:));
Hinv=inv(dfdWdP'*dfdWdP); % inverse of Equation (20)
f_bar=mean(f(:)); f_tilde=sqrt(sum((f(:)-f_bar).^2));
flag=0; iter=0; dP=ones(size(P));
while flag==0
    [dXY]=W(dX(:),dY(:),P); % Equation (14)
    g=InterpCoef(Xos(2).*ones(size(dXY,1),1)+dXY(:,2),
    Xos(1).*ones(size(dXY,1),1)+dXY(:,1));
    g_bar=mean(g(:)); g_tilde=sqrt(sum((g(:)-g_bar).^2));
    StopVal=StopCrit(dP,(SubSize-1)/2); % Equation (23)
    if any([StopVal<StopCritVal,iter>100])
      flag=1;
      C=1-sum(((f(:)-f_bar)/f_tilde-(g(:)-g_bar)/g_tilde).^2)/2;
      % Equation (11) substituted into Equation (13)
    else
      J=dfdWdP'*(f(:)-f_bar-f_tilde/g_tilde*(g(:)-g_bar));
      % Summation of Equation (19)
      dP([1:SFOrder*3+0^SFOrder 7:6+SFOrder*3+0^SFOrder])= -Hinv*J; % Equation (19)
      P=Mat2SFPVec(SFPVec2Mat(P)/SFPVec2Mat(dP)); % Equation (21)
    end
    iter=iter+1;
  end
```

```
function [W,dFdWdP,SFPVec2Mat,Mat2SFPVec,StopCrit]=
  SFExpressions(SFOrder)
  switch SFOrder
  case 0 % Zero order SF
    W=@(dX,dY,P) [P(1)+dX,P(7)+dY];  % Equation (14)
    dFdWdP=@(dX,dY,dfdx,dfdy) [dfdx,dfdy];
    SFPVec2Mat=@(P) reshape([1,0,0,0,1,0,P(1),P(7),1],[3,3]); % Equation (22)
    Mat2SFPVec=@(W) [W(7),0,0,0,0,0,W(8),0,0,0,0,0];
    StopCrit=@(dP,Zeta) sqrt(sum((dP'.*[1,0,0,0,0,0,1,0,0,0,0,0])
    .^2)); % Equation (23)
  case 1 % First order SF
    W=@(dX,dY,P) [P(1)+P(3).*dY+dX.*(P(2)+1),
    P(7)+P(8).*dX+dY.*(P(9)+1)]; % Equation (14)
    dFdWdP=@(dX,dY,dfdx,dfdy) [dfdx,dfdx.*dX,dfdx.
    *dY, dfdy,dfdy.*dX,dfdy.*dY];
    SFPVec2Mat=@(P) reshape([P(2)+1,P(8),0,P(3),P(9)+1,0,P(1),
    P(7),1],[3,3]); % Equation (22)
    Mat2SFPVec=@(W) [W(7),W(1)-1.0,W(4),0,0,0,W(8),W(2),
    W(5)-1.0,0,0,0];
    StopCrit=@(dP,Zeta) sqrt(sum((dP'.*[1,Zeta,Zeta,0,0,0,1,Zeta,
    Zeta,0,0,0]).^2)); % Equation (23)
  case 2 % Second order SF
    W=@(dX,dY,P) [P(1)+P(3).*dY+P(4).*dX.^2.*(1/2)+P(6).*dY.^2.*
    (1/2)+dX.*(P(2)+1)+P(5).*dX.*dY,P(7)+P(8).*dX+P(10).*dX.^2.*(1/2)+
    P(12).*dY.^2.*(1/2)+dY.*(P(9)+1)+P(11).*dX.*dY]; % Equation (14)
    dFdWdP=@(dX,dY,dfdx,dfdy) [dfdx,dfdx.*dX,dfdx.*dY,
    (dfdx.*dX.^2)/2,dfdx.*dX.*dY,(dfdx.*dY.^2)/2,dfdy,dfdy.*dX,dfdy.
    *dY,(dfdy.*dX.^2)/2,dfdy.*dX.*dY,(dfdy.*dY.^2)/2];
    SFPVec2Mat=@(P) reshape([P(2)*2+P(1)*P(4)+P(2)^2+1,
    P(1)*P(10)*1/2+P(4)*P(7)*(1/2)+P(8)*(P(2)*2+2)*1/2,P(7)*P(10)+P(8)^2,
    P(4)*1/2,P(10)*1/2,0,P(1)*P(5)*2+P(3)*(P(2)*2+2),P(2)+P(9)+P(2)*P(9)+
    P(3)*P(8)+P(1)*P(11)+P(5)*P(7)+1,P(7)*P(11)*2.0+P(8)*(P(9)+1)*2,P(5),
    P(11),0,P(1)*P(6)+P(3)^2,P(1)*P(12)*1/2+P(6)*P(7)*1/2+P(3)*(P(9)+1),
    P(9)*2+P(7)*P(12)+P(9)^2+1,P(6)*1/2,P(12)*1/2,0,P(1)*(P(2)+1)*2,P(7)+
    P(1)*P(8)+P(2)*P(7),P(7)*P(8)*2,P(2)+1,P(8),0,P(1)*P(3)*2,P(1)+P(1)*P(9)+
    P(3)*P(7),P(7)*(P(9)+1)*2,P(3),P(9)+1,0,P(1)^2,P(1)*P(7),P(7)^2,P(1),
    P(7),1],[6,6]); % Equation (22)
    Mat2SFPVec=@(W) [W(34),W(22)-1,W(28),W(4).*2,W(10),W(16).*2,
    W(35),W(23),W(29)-1,W(5).*2,W(11),W(17).*2];
    StopCrit=@(dP,Zeta) sqrt(sum((dP'.*[1,Zeta,Zeta,0.5*Zeta,
    0.5*Zeta,0.5*Zeta,1,Zeta,Zeta,0.5*Zeta,0.5*Zeta,0.5*Zeta]).^2)); % Equation (23)
  end
```

```
function [u,v]=PCM(F,G,SubSize,XosX,XosY,SubExtract)
 NCPS=(fft2(SubExtract(F,[XosX,XosY],SubSize))
 .*conj(fft2(SubExtract(G,[XosX,XosY],SubSize))))./abs(fft2(SubExtract(
 F,[XosX,XosY],SubSize)).*conj(fft2(SubExtract(G,[XosX,XosY],SubSize))));
 CC=abs(ifft2(NCPS));
 [vid,uid]=find(CC==max(CC(:)));
 IndShift=-ifftshift(-fix(SubSize/2):ceil(SubSize/2)-1);
 u=IndShift(uid);
 v=IndShift(vid);
```

```
function PD=CSTrans(n,PD,WorldCTs,ImgCTs,rho)
```

```
if WorldCTs~=0 | ImgCTs~=0
    CamParams=estimateCameraParameters(ImgCTs,WorldCTs); % Section 2.1.4
else
    CamParams=cameraParameters('ImageSize',PD(1).ImgSize,
    'IntrinsicMatrix',eye(3),'WorldPoints',PD(1).Xos','RotationVectors',
    [0,0,0],'TranslationVectors',[0,0,0]);
    WorldCTs=PD(1).Xos'; ImgCTs=PD(1).Xos';
end
[R,T]=extrinsics(ImgCTs(:,:,end),WorldCTs,CamParams);
Tspec=T-[0,0,rho]*R; % Equation (6)
for d=1:n
    Xds(1,:)=PD(d).Xos(1,:)+PD(d).P(1,:); % Equation (24)
    Xds(2,:)=PD(d).Xos(2,:)+PD(d).P(7,:); % Equation (24)
    PD(d).Xow=pointsToWorld(CamParams,R,Tspec, undistortPoints(PD(d).Xos',CamParams))';
    Xdw=pointsToWorld(CamParams,R,Tspec, undistortPoints(Xds',CamParams))';
    PD(d).Uw=Xdw-PD(d).Xow; % Equation (26)
    PD(d).CamParams=CamParams;
end
```

References

1. Pan, B.; Qian, K.; Xie, H.; Asundi, A. Two-dimensional digital image correlation for in-plane displacement and strain measurement: A review. *Meas. Sci. Technol.* **2009**, *20*, 062001. [CrossRef]
2. Pan, B. Digital image correlation for surface deformation measurement: Historical developments, recent advances and future goals. *Meas. Sci. Technol.* **2018**, *29*, 082001. [CrossRef]
3. Shao, X.; Dai, X.; Chen, Z.; He, X. Real-time 3D digital image correlation method and its application in human pulse monitoring. *Appl. Opt.* **2016**, *55*, 696–704. [CrossRef]
4. Lin, D.; Zhang, A.; Gu, J.; Chen, X.; Wang, Q.; Yang, L.; Chou, Y.; Liu, G.; Wang, J. Detection of multipoint pulse waves and dynamic 3D pulse shape of the radial artery based on binocular vision theory. *Comput. Methods Programs Biomed.* **2018**, *155*, 61–73. [CrossRef] [PubMed]
5. Tuononen, A.J. Digital Image Correlation to analyse stick-slip behaviour of tyre tread block. *Tribol. Int.* **2014**, *69*, 70–76. [CrossRef]
6. Sutton, M.A.; Ke, X.; Lessner, S.M.; Goldbach, M.; Yost, M.; Zhao, F.; Schreier, H.W. Strain field measurements on mouse carotid arteries using microscopic three-dimensional digital image correlation. *J. Biomed. Mater. Res. Part A* **2008**, *84*, 178–190. [CrossRef]
7. Palanca, M.; Tozzi, G.; Cristofolini, L. The use of digital image correlation in the biomechanical area: A review. *Int. Biomech.* **2016**, *3*, 1–21. [CrossRef]
8. Zhang, D.; Arola, D.D. Applications of digital image correlation to biological tissues. *J. Biomed. Opt.* **2004**, *9*, 691–700. [CrossRef]
9. Winkler, J.; Hendy, C. Improved structural health monitoring of London's Docklands Light Railway bridges using Digital image correlation. *Struct. Eng. Int.* **2017**, *27*, 435–440. [CrossRef]
10. Reagan, D.; Sabato, A.; Niezrecki, C. Feasibility of using digital image correlation for unmanned aerial vehicle structural health monitoring of bridges. *Struct. Health Monit.* **2018**, *17*, 1056–1072. [CrossRef]
11. LeBlanc, B.; Niezrecki, C.; Avitabile, P.; Chen, J.; Sherwood, J. Damage detection and full surface characterization of a wind turbine blade using three-dimensional digital image correlation. *Struct. Health Monit.* **2013**, *12*, 430–439. [CrossRef]
12. Molina-Viedma, A.J.; Felipe-Sesé, L.; López-Alba, E.; Díaz, F.A. 3D mode shapes characterisation using phase-based motion magnification in large structures using stereoscopic DIC. *Mech. Syst. Signal Process.* **2018**, *108*, 140–155. [CrossRef]
13. Barone, S.; Neri, P.; Paoli, A.; Razionale, A.V. Low-frame-rate single camera system for 3D full-field high-frequency vibration measurements. *Mech. Syst. Signal Process.* **2019**, *123*, 143–152. [CrossRef]
14. Bickel, V.T.; Manconi, A.; Amann, F. Quantitative assessment of digital image correlation methods to detect and monitor surface displacements of large slope instabilities. *Remote Sens.* **2018**, *10*, 865. [CrossRef]
15. Walter, T.R.; Legrand, D.; Granados, H.D.; Reyes, G.; Arámbula, R. Volcanic eruption monitoring by thermal image correlation: Pixel offsets show episodic dome growth of the Colima volcano. *J. Geophys. Res. Solid Earth* **2013**, *118*, 1408–1419. [CrossRef]

16. Leprince, S.; Barbot, S.; Ayoub, F.; Avouac, J.P. Automatic and precise orthorectification, coregistration, and subpixel correlation of satellite images, application to ground deformation measurements. *IEEE Trans. Geosci. Remote Sens.* **2007**, *45*, 1529–1558. [CrossRef]
17. Heid, T.; Kääb, A. Evaluation of existing image matching methods for deriving glacier surface displacements globally from optical satellite imagery. *Remote Sens. Environ.* **2012**, *118*, 339–355. [CrossRef]
18. Wang, P.; Pierron, F.; Thomsen, O.T. Identification of material parameters of PVC foams using digital image correlation and the virtual fields method. *Exp. Mech.* **2013**, *53*, 1001–1015. [CrossRef]
19. Grédiac, M.; Pierron, F.; Surrel, Y. Novel procedure for complete in-plane composite characterization using a single T-shaped specimen. *Exp. Mech.* **1999**, *39*, 142–149. [CrossRef]
20. Huchzermeyer, R. Measuring Mechanical Properties Using Digital Image Correlation: Extracting Tensile and Fracture Properties from A Single Sample. Master's Thesis, Stellenbosch University, Stellenbosch, South Africa, 2017.
21. Leclerc, H.; Périé, J.N.; Roux, S.; Hild, F. Integrated digital image correlation for the identification of mechanical properties. In *Proceedings of the International Conference on Computer Vision/Computer Graphics Collaboration Techniques and Applications, Rocquencourt, France, 4–6 May 2009*; Gagalowic, A., Philips, W., Eds.; Springer: Heidelberg/Berlin, Germany, 2009; pp. 161–171.
22. Chevalier, L.; Calloch, S.; Hild, F.; Marco, Y. Digital image correlation used to analyze the multiaxial behavior of rubber-like materials. *Eur. J. Mech. A Solids* **2001**, *20*, 169–187. [CrossRef]
23. van Rooyen, M.; Becker, T.H. High-temperature tensile property measurements using digital image correlation over a non-uniform temperature field. *J. Strain Anal. Eng. Des.* **2018**, *53*, 117–129. [CrossRef]
24. Peters, W.H.; Ranson, W.F. Digital imaging techniques in experimental stress analysis. *Opt. Eng.* **1982**, *21*, 427–431. [CrossRef]
25. Peters, W.H.; Ranson, W.F.; Sutton, M.A.; Chu, T.C.; Anderson, J. Application of digital correlation methods to rigid body mechanics. *Opt. Eng.* **1983**, *22*, 738–742. [CrossRef]
26. Chu, T.C.; Ranson, W.F.; Sutton, M.A. Applications of digital-image-correlation techniques to experimental mechanics. *Exp. Mech.* **1985**, *25*, 232–244. [CrossRef]
27. Sutton, M.; Mingqi, C.; Peters, W.; Chao, Y.; McNeill, S. Application of an optimized digital correlation method to planar deformation analysis. *Image Vis. Comput.* **1986**, *4*, 143–150. [CrossRef]
28. Bruck, H.A.; McNeill, S.R.; Sutton, M.A.; Peters, W.H. Digital image correlation using newton-raphson method of partial differential correction. *Exp. Mech.* **1989**, *29*, 261–267. [CrossRef]
29. Luo, P.F.; Chao, Y.J.; Sutton, M.A.; Peters, W.H. Accurate measurement of three-dimensional deformations in deformable and rigid bodies using computer vision. *Exp. Mech.* **1993**, *33*, 123–132. [CrossRef]
30. Bay, B.K.; Smith, T.S.; Fyhrie, D.P.; Saad, M. Digital volume correlation: Three-dimensional strain mapping using X-ray tomography. *Exp. Mech.* **1999**, *39*, 217–226. [CrossRef]
31. Schreier, H.W.; Braasch, J.R.; Sutton, M.A. Systematic errors in digital image correlation caused by intensity interpolation. *Opt. Eng.* **2000**, *39*, 2915–2921. [CrossRef]
32. Lu, H.; Cary, P.D. Deformation measurements by digital image correlation: Implementation of a second-order displacement gradient. *Exp. Mech.* **2000**, *40*, 393–400. [CrossRef]
33. Baker, S.; Matthews, I. Lucas-kanade 20 years on: A unifying framework. *Int. J. Comput. Vis.* **2004**, *56*, 221–255. [CrossRef]
34. Tong, W. An evaluation of digital image correlation criteria for strain mapping applications. *Strain* **2005**, *41*, 167–175. [CrossRef]
35. Pan, B.; Li, K.; Tong, W. Fast, robust and accurate digital image correlation calculation without redundant computations. *Exp. Mech.* **2013**, *53*, 1277–1289. [CrossRef]
36. Gao, Y.; Cheng, T.; Su, Y.; Xu, X.; Zhang, Y.; Zhang, Q. High-efficiency and high-accuracy digital image correlation for three-dimensional measurement. *Opt. Lasers Eng.* **2015**, *65*, 73–80. [CrossRef]
37. Pan, B.; Wang, B. Digital image correlation with enhanced accuracy and efficiency: A comparison of two subpixel registration algorithms. *Exp. Mech.* **2016**, *56*, 1395–1409. [CrossRef]
38. Blaber, J.; Adair, B.; Antoniou, A. Ncorr: Open-source 2D digital image correlation matlab software. *Exp. Mech.* **2015**, *55*, 1105–1122. [CrossRef]

39. Reu, P.L.; Toussaint, E.; Jones, E.; Bruck, H.A.; Iadicola, M.; Balcaen, R.; Turner, D.Z.; Siebert, T.; Lava, P.; Simonsen, M. DIC challenge: Developing images and guidelines for evaluating accuracy and resolution of 2D analyses. *Exp. Mech.* **2018**, *58*, 1067–1099. [CrossRef]
40. Bloomenthal, J.; Rokne, J. Homogeneous coordinates. *Vis. Comput.* **1994**, *11*, 15–26. [CrossRef]
41. Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 1330–1334. [CrossRef]
42. Heikkila, J.; Silven, O. Four-step camera calibration procedure with implicit image correction. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Juan, PR, USA, 17–19 June 1997.
43. Tsai, R.Y. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE J. Robot. Autom.* **1987**, *3*, 323–344. [CrossRef]
44. Wei, G.Q.; De Ma, S. Implicit and explicit camera calibration: Theory and experiments. *IEEE Trans. Pattern Anal. Mach. Intell.* **1994**, *16*, 469–480. [CrossRef]
45. Pan, B.; Xie, H.; Wang, Z. Equivalence of digital image correlation criteria for pattern matching. *Appl. Opt.* **2010**, *49*, 5501–5509. [CrossRef]
46. Keys, R.G. Cubic convolution interpolation for digital image processing. *IEEE Trans. Acoust.* **1981**, *29*, 1153–1160. [CrossRef]
47. Hou, H.S.; Andrews, H.C. Cubic splines for image interpolation and digital filtering. *IEEE Trans. Acoust.* **1978**, *26*, 508–517. [CrossRef]
48. Pan, Z.; Chen, W.; Jiang, Z.; Tang, L.; Liu, Y.; Liu, Z. Performance of global look-up table strategy in digital image correlation with cubic B-spline interpolation and bicubic interpolation. *Theor. Appl. Mech. Lett.* **2016**, *6*, 126–130. [CrossRef]
49. Pan, B. Bias error reduction of digital image correlation using gaussian pre-filtering. *Opt. Lasers Eng.* **2013**, *51*, 1161–1167. [CrossRef]
50. Pan, B.; Xie, H.; Wang, Z.; Qian, K.; Wang, Z. Study on subset size selection in digital image correlation for speckle patterns. *Opt. Express* **2008**, *16*, 7037–7048. [CrossRef]
51. Żołądek, H. The topological proof of abel-ruffini theorem. *Topol. Methods Nonlinear Anal.* **2000**, *16*, 253–265. [CrossRef]
52. Pan, B.; Asundi, A.; Xie, H.; Gao, J. Digital image correlation using iterative least squares and pointwise least squares for displacement field and strain field measurements. *Opt. Lasers Eng.* **2009**, *47*, 865–874. [CrossRef]
53. Zhou, Y.; Sun, C.; Chen, J. Adaptive subset offset for systematic error reduction in incremental digital image correlation. *Opt. Lasers Eng.* **2014**, *55*, 5–11. [CrossRef]
54. Foroosh, H.; Zerubia, J.B.; Berthod, M. Extension of phase correlation to subpixel registration. *IEEE Trans. Image Process.* **2002**, *11*, 188–200. [CrossRef] [PubMed]
55. Orteu, J.-J.; Garcia, D.; Robert, L.; Bugarin, F. A speckle texture image generator. In Proceedings of the Speckle06: Speckles, From Grains to Flowers, Nîmes, France, 13–15 September 2006; Slangen, P., Cerruti, C., Eds.; SPIE: Bellingham, WA, USA, 2006; pp. 104–109.
56. Reu, P.L. Experimental and numerical methods for exact subpixel shifting. *Exp. Mech.* **2011**, *51*, 443–452. [CrossRef]
57. Bornert, M.; Brémand, F.; Doumalin, P.; Dupré, J.C.; Fazzini, M.; Grédiac, M.; Hild, F.; Mistou, S.; Molimard, J.; Orteu, J.J.; et al. Assessment of digital image correlation measurement errors: Methodology and results. *Exp. Mech.* **2009**, *49*, 353–370. [CrossRef]
58. Amiot, F.; Bornert, M.; Doumalin, P.; Dupré, J.C.; Fazzini, M.; Orteu, J.J.; Poilâne, C.; Robert, L.; Rotinat, R.; Toussaint, E.; et al. Assessment of digital image correlation measurement accuracy in the ultimate error regime: Main results of a collaborative benchmark. *Strain* **2013**, *49*, 483–496. [CrossRef]
59. Harilal, R.; Ramji, M. Adaptation of open source 2D DIC software ncorr for solid mechanics applications. In Proceedings of the 9th International Symposium on Advanced Science and Technology in Experimental Mechanics, New Delhi, India, 1–6 November 2014.
60. Lunt, D.; Thomas, R.; Roy, M.; Duff, J.; Atkinson, M.; Frankel, P.; Preuss, M.; Quinta da Fonseca, J. Comparison of sub-grain scale digital image correlation calculated using commercial and open-source software packages. *Mater. Charact.* **2020**, *163*, 110271. [CrossRef]

61. Pan, B. Reliability-guided digital image correlation for image deformation measurement. *Appl. Opt.* **2009**, *48*, 1535–1542. [CrossRef] [PubMed]
62. Schreier, H.W.; Sutton, M.A. Systematic errors in digital image correlation due to undermatched subset shape functions. *Exp. Mech.* **2002**, *42*, 303–310. [CrossRef]

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).