applied sciences

Article

Stereo Digital Image Correlation in MATLAB

Devan Atkinson and Thorsten Hermann Becker *

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

Abstract: Digital Image Correlation (DIC) has found widespread use in measuring full-field displacements and deformations experienced by a body from images captured of it. Stereo-DIC has received significantly more attention than two-dimensional (2D) DIC since it can account for out-of-plane displacements. Although many aspects of Stereo-DIC that are shared in common with 2D DIC are well documented, there is a lack of resources that cover the theory of Stereo-DIC. Furthermore, publications which do detail aspects of the theory do not detail its implementation in practice. This literature gap makes it difficult for newcomers to the field of DIC to gain a deep understanding of the Stereo-DIC process, although this knowledge is necessary to contribute to the development of the field by either furthering its capabilities or adapting it for novel applications. This gap in literature acts as a barrier thereby limiting the development rate of Stereo-DIC. This paper attempts to address this by presenting the theory of a subset-based Stereo-DIC framework that is predominantly consistent with the current state-of-the-art. The framework is implemented in practice as a 202 line MATLAB code. Validation of the framework shows that it performs on par with well-established Stereo-DIC algorithms, indicating it is sufficiently reliable for practical use. Although the framework is designed to serve as an educational resource, its modularity and validation make it attractive as a means to further the capabilities of DIC.

Keywords: Digital Image Correlation; stereo; subset-based; education; MATLAB code; three-dimensional

Citation: Atkinson, D.; Becker, T.H. Stereo Digital Image Correlation in MATLAB. *Appl. Sci.* **2021**, *11*, 4904. https://doi.org/10.3390/app11114904

Academic Editor: Andrés Márquez

Received: 6 May 2021
Accepted: 17 May 2021
Published: 26 May 2021

1. Introduction

Digital Image Correlation (DIC) is an optical metrology technique capable of determining full-field displacements and deformations experienced by a body from images captured of the body's surface. The accuracy of DIC results is greatly influenced by the quality of the speckle pattern [1], the internals of the DIC algorithm [2,3], the appropriateness of the chosen processing parameters [4] and the quality of the calibration process [5–7], among others. However, despite this, DIC, a non-interferometric technique, is more robust in withstanding vibrations and ambient light variations than interferometric techniques, such as Electronic Speckle Pattern Interferometry and Moiré interferometry, allowing it to be used outside the confines of a laboratory. Additionally, its non-contact, full-field nature; its ability to function at a range of scales (from scanning electron microscopy images [8,9] to satellite images [10]); and the rapid improvement of cameras has resulted in it becoming one of the most favored techniques in experimental solid mechanics.

Although DIC has received the most attention and thus development within experimental solid mechanics, it has found widespread use across many fields. Some applications of DIC include: (i) investigating mechanical properties of biological tissues [11–15]; (ii) deflection measurement and structural health monitoring of bridges [16–22]; (iii) investigating cracks in masonry structures [23–25]; (iv) landslide monitoring [26–29]; (v) determining material properties of woven fabrics (such as cotton) [30,31]; vibrational analysis of components [32–34]; and (vii) in situ damage detection of components and structures [35–37] to name a few.

Following the introduction of two-dimensional (2D) DIC in 1982 by Peters and Ranson [38] it was extended to three dimensions (Stereo-DIC) by Luo et al. [39] in 1993 and to digital volume correlation (DVC) by Bay et al. [40] in 1999. The correlation aspect of DIC has undergone significant refinement over the years. Current state-of-the-art DIC algorithms as identified by Pan [41] make use of: (i) bi-quintic b-spline interpolation which was shown by Schreier et al. [3] to produce the most accurate sub-pixel displacements; (ii) the first and second-order shape functions (SFs) proposed by Peters et al. [42] and Lu and Carry [43] respectively; (iii) the zero-mean normalized sum of squared difference (ZNSSD) correlation criterion suggested by Tong [44]; (iv) Gaussian pre-filtering of images to reduce bias in the displacements caused by high frequency noise in the images as proposed by Pan [45]; and (v) the inverse compositional Gauss–Newton (IC-GN) optimization method introduced by Baker and Matthews [46]. Although the IC-GN optimization method is theoretically equivalent to the Newton–Raphson method [46], in practice it has greater efficiency, accuracy and robustness to withstand noise [47] making it preferable.

Although DIC is a complex process, the papers proposed by Pan et al. [48], Gao et al. [49], Solav et al. [50], and Blaber et al. [51] provide an in-depth explanation of the theory of the DIC process, which coupled with the Good Practices Guide for DIC [52], enables newcomers to the field of DIC to gain the necessary knowledge to effectively apply it in established use cases. However, in order for a newcomer to contribute to the field of DIC by developing new algorithms which improve its capabilities or adapting it for new applications, he or she must gain a deep understanding of DIC. Gaining such a deep understanding is cumbersome due to the lack of resources which directly bridge the gap between the theory of DIC and its coded implementation. More specifically, although some publications release code that is consistent with the theory presented [50,51], these codes focus on robustness and ease of use which, despite making them more suitable for real world applications, results in a complex code which is ineffective as a learning resource. This lack of resources acts as a barrier to newcomers intending to further the capabilities of DIC, thereby limiting the development rate of the field.

It is for this reason that the authors published a paper bridging the gap between the theory of a 2D DIC framework (ADIC2D) and its practical implementation as a modular 117 line MATLAB code [53]. However, 2D DIC, being limited to determining in-plane displacements, is susceptible to errors in the presence of out-of-plane motion, which is generally unavoidable especially outside the confines of a laboratory [54]. As such, Stereo-DIC has become favored since it accounts for out-of-plane motion, allowing for more reliable use across a broader range of applications and fields. This is supported by the significantly larger number of publications of Stereo-DIC relative to 2D DIC.

This paper builds upon the work of the preceding paper by extending ADIC2D to Stereo-DIC with the aim of bridging the gap between the theory and implementation of Stereo-DIC. This is done by first presenting the theory of a subset-based, Stereo-DIC framework that is predominantly consistent with current state-of-the-art techniques. How this theory is implemented as a modular 202 line MATLAB code is then discussed. These theory and implementation sections focus on aspects of Stereo-DIC which differ from that of ADIC2D. Thereafter the framework is validated using image sets provided by the Stereo-DIC Challenge [55]. More specifically, ADIC3D's results are compared to those of LaVision's StrainMaster software and DICe [56] in order to make observations about its capabilities. Finally, the proposed framework's modularity, limitations, and overall performance are discussed.

The framework, referred to as ADIC3D, was developed in MATLAB since its simple syntax lets the reader focus on the core mathematics of the code. Furthermore, MATLAB's efficient built-in functions are leveraged to simplify the code and reduce its computation time. The code is designed to be modular, so that the link between the theory and code is clear and enables readers to systematically develop an understanding of the code. Additionally, this modularity allows the code to be easily altered, urging readers to further the capabilities of DIC.

2. Framework Theory

Stereo-DIC is fundamentally an extension of 2D DIC, which relies upon two or more cameras simultaneously capturing images of the specimen (where specimen refers to the body being tracked) from different perspectives in order to determine its in-plane and out-of-plane displacements. As such, there are aspects of Stereo-DIC which are identical to those of 2D DIC; namely calibration and correlation. Stereo-DIC relies upon epipolar geometry to determine in-plane and out-of-plane displacements in the real world from in-plane displacements determined within images. Section 2.5 discusses how calibration and correlation are employed by Stereo-DIC based on epipolar geometry and how triangulation methods are used to perform displacement transformations.

Although Stereo-DIC can employ more than two cameras [50], ADIC3D is limited to the conventional two-camera implementation. Variables that need to be defined on a per camera basis have the subscript j to indicate that they are associated with the j^{th} camera (i.e., camera 1 and 2). Within this work, an image series is defined as the collection of images captured by a single camera and an image set refers to the two corresponding image series. Thus, an image pair refers to the two images, one from each image series, which were captured simultaneously.

2.1. Homogeneous Coordinates

Calibration and triangulation use homogeneous coordinates [57], to represent coordinates in projective space, such that affine and projective transformations can be implemented through matrix multiplication. Appending a scaling variable of unity to a n dimensional vector in Euclidean space converts it to an $n+1$ dimensional vector in projective space. Converting back to Euclidean space, represented as function Φ, entails dividing the vector's elements by the scaling variable before eliminating it to reduce the vector to n dimensions. Underlined variables denote homogenous coordinate vectors.

2.2. Calibration

Calibration, explained in terms of a single camera since each camera is calibrated separately, determines the parameters of the camera model, which involves the four coordinate systems (CSs) depicted in Figure 1: (i) the three-dimensional (3D) world CS having arbitrary orientation and position; (ii) the 3D camera CS with its origin at the aperture of the camera and its z-axis aligned with the optical axis; (iii) the 2D ideal sensor CS which is coincident with the plane of the charge coupled device (CCD) representing the image in the absence of distortion; and (iv) the 2D distorted sensor CS which is coincident with the ideal sensor CS but represents the actual image by accounting for radial distortion. No symbols have been assigned to the camera CS since it is not directly involved in DIC.

The camera model uses the pinhole camera model to project a 3D coordinate of a point on the specimen in the world CS to its corresponding 2D location in the ideal sensor CS before the radial distortion model determines its location in the distorted sensor CS. Although this camera model, and subsequent calibration process, is rather simple, it performs sufficiently well as shown in Section 4. However, the accuracy the displacements determined could be improved by using a more sophisticated camera calibration method which provides a more complete description of the camera optics such as the generic camera calibration method [58,59].

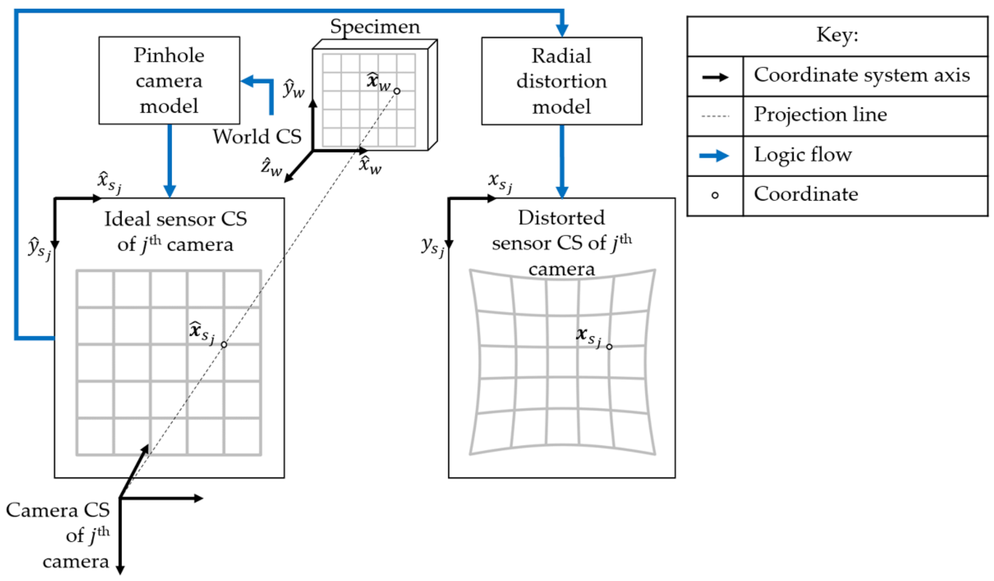

Figure 1. Schematic diagram of coordinate systems employed by the camera model.

2.2.1. Pinhole Camera Model

The projection matrix $\boldsymbol{Q}_j$, consisting of intrinsic ($\boldsymbol{K}_j$) and extrinsic ($\boldsymbol{V}_j$) parameter matrices, relates a coordinate in the world CS, $\underline{\hat{\boldsymbol{x}}}_w = [\hat{x}_w \quad \hat{y}_w \quad \hat{z}_w \quad 1]^T$, to a coordinate in the ideal sensor CS, $\underline{\hat{\boldsymbol{x}}}_{s_j} = \left[\hat{x}_{s_j} \quad \hat{y}_{s_j} \quad 1\right]^T$, as

$$\alpha_j \underline{\hat{\boldsymbol{x}}}_{s_j} = \boldsymbol{Q}_j \underline{\hat{\boldsymbol{x}}}_w = \boldsymbol{K}_j \boldsymbol{V}_j \underline{\hat{\boldsymbol{x}}}_w = \begin{bmatrix} \xi_{x_j} & c_{s_j} & c_{x_j} \\ 0 & \xi_{y_j} & c_{y_j} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_j^{11} & R_j^{12} & R_j^{13} & T_j^1 \\ R_j^{21} & R_j^{22} & R_j^{23} & T_j^2 \\ R_j^{31} & R_j^{32} & R_j^{33} & T_j^3 \end{bmatrix} \underline{\hat{\boldsymbol{x}}}_w. \tag{1}$$

Here α_j is the arbitrary scaling variable of the homogenous coordinates. First, the rotation matrix, $\boldsymbol{R}_j$, and translation vector, $\boldsymbol{T}_j$, transform $\underline{\hat{\boldsymbol{x}}}_w$ to the camera CS. Thereafter this 3D coordinate is projected to the ideal sensor CS using $\boldsymbol{K}_j$ which performs four operations: (i) projects the 3D coordinate to a 2D coordinate on the plane of the CCD; (ii) scales the coordinate from metric units to units of pixels using ratios ξ_{x_j} and ξ_{y_j} in the x- and y-directions respectively; (iii) applies a translation, c_{x_j} and c_{y_j} in the x- and y-directions respectively, placing the origin of the ideal sensor CS at the top left of the image; and (iv) converts from the orthogonal camera CS to a skew ideal sensor CS using c_{s_j}. In this work $c_{s_j} = 0$ since an orthogonal ideal sensor CS is assumed.

$\boldsymbol{K}_j$ is only dependent on the camera system and does not change if the camera position changes. In contrast, $\boldsymbol{V}_j$ depends on the relative position and orientation between the camera and the world CS.

2.2.2. Radial Distortion Model

The radial distortion model is sufficient to account for distortion in the images caused by imperfections of the lens system [60]. First the normalized, ideal image coordinates $\hat{\boldsymbol{x}}_{n_j} = \left[\hat{x}_{n_j} \quad \hat{y}_{n_j}\right]^T$ are computed as

$$\hat{\boldsymbol{x}}_{n_j} = \begin{bmatrix} \hat{x}_{n_j} \\ \hat{y}_{n_j} \end{bmatrix} = \Phi\left(\boldsymbol{K}_j^{-1} \underline{\hat{\boldsymbol{x}}}_{s_j}\right). \tag{2}$$

Thereafter, the unit-less radial distortion parameters, κ_j^1 and κ_j^2, are used to compute the normalized, distorted image coordinates, $\boldsymbol{x}_{n_j} = [x_{n_j} \quad y_{n_j}]^T$, as [61]

$$\boldsymbol{x}_{n_j} = \left(1 + \kappa_j^1 \widehat{\boldsymbol{x}}_{n_j}^T \widehat{\boldsymbol{x}}_{n_j} + \kappa_j^2 \left(\widehat{\boldsymbol{x}}_{n_j}^T \widehat{\boldsymbol{x}}_{n_j}\right)^2\right) \widehat{\boldsymbol{x}}_{n_j}. \tag{3}$$

Finally, the distorted coordinates in the distorted sensor CS, $\boldsymbol{x}_{s_j} = [x_{s_j} \quad y_{s_j}]^T$, are obtained as

$$\boldsymbol{x}_{s_j} = \begin{bmatrix} x_{s_j} \\ y_{s_j} \end{bmatrix} = \Phi\left(\boldsymbol{K}_j \begin{bmatrix} x_{n_j} \\ y_{n_j} \\ 1 \end{bmatrix}\right). \tag{4}$$

2.2.3. Determining Calibration Parameters

Prior to performing calibration, a calibration image series must be captured of the calibration plate. The calibration plate is a planar object containing distinct, point-like features, called calibration targets (CTs), at known 3D locations which define the position and orientation of the world CS. A corresponding set of 2D coordinates in the distorted sensor CS are obtained by locating the CTs in the calibration images. These sets of 3D and 2D coordinates are used to solve for the parameters of the camera model, which describe the relationship between the two. Using at least 50 calibration images is recommended since this inverse problem is sensitive to noise in the 3D and 2D coordinates [41,62].

Calibration is performed using two steps. First, initial estimates for $\boldsymbol{K}_j$ and $\boldsymbol{V}_j$ are obtained using the method proposed by Zhang [61] while κ_j^1 and κ_j^2 are set to zero. Secondly, non-linear, least-squares optimization is used to optimize all the parameters by minimizing the total projection error, E_{proj}, given as

$$E_{proj} = \sum_{l=1}^{L} \left(\left(x_l^{calc} - x_l^{true}\right)^2 + \left(y_l^{calc} - y_l^{true}\right)^2\right). \tag{5}$$

Here $\boldsymbol{x}_l^{true} = [x_l^{true} \quad y_l^{true}]^T$ and $\boldsymbol{x}_l^{calc} = [x_l^{calc} \quad y_l^{calc}]^T$ are the locations of the lth CT in the distorted sensor CS determined from the calibration image series and predicted by the camera model respectively. There is a total of L many CTs visible within the calibration image series.

2.3. Correlation

ADIC3D uses correlation to track a subset, a cluster of pixels, between images. The specimen's surface requires an isotropic, random pattern with high information content such that subsets are tracked reliably. Correlation operates on two images: a reference image, F, representing the specimen in its relaxed state and a deformed image, G, representing the specimen after experiencing displacement and/or deformation. Correlation determines how a subset of the reference image, called a reference subset ($\boldsymbol{f}$), must displace and deform such that it matches a corresponding subset in the deformed image, called an investigated subset ($\boldsymbol{g}$). This is referred to as the correspondence problem, which is mathematically defined as the objective function. For a more detailed discussion on the correspondence problem, the reader is referred to the authors' previous paper [53].

Note that correlation operates solely within the distorted sensor CS and is unconcerned with which camera the distorted sensor CS belongs to. Thus, in Sections 2.3 and 3.2, the subscript s_j is dropped from $\boldsymbol{x}_{s_j}$ (i.e., $\boldsymbol{x}$) to simplify the mathematics.

To understand how the objective function is comprised of the SF, the interpolation method and correlation criterion first consider that the ith pixel position of $\boldsymbol{f}$, $\boldsymbol{x}_i = [x_i \quad y_i]^T$, is represented by the reference subset center position, $\boldsymbol{x}^o = [x^o \quad y^o]^T$, and the distance from $\boldsymbol{x}^o$ to $\boldsymbol{x}_i$, $\Delta\boldsymbol{x}_i = [\Delta x_i \quad \Delta y_i]^T$, as

$$\boldsymbol{x}_i = \Delta\boldsymbol{x}_i + \boldsymbol{x}^o. \tag{6}$$

The SF, $\boldsymbol{W}$, determines how the pixel positions of $\boldsymbol{f}$ displace as a result of the displacement and deformation of $\boldsymbol{f}$, which is quantified by the shape function parameters (SFPs), $\boldsymbol{P}$. The SF does this by modifying $\Delta\boldsymbol{x}_i$ as

$$\Delta \boldsymbol{x}_i' = \begin{bmatrix} \Delta x_i' \\ \Delta y_i' \end{bmatrix} = \boldsymbol{W}(\Delta \boldsymbol{x}_i, \boldsymbol{P}). \tag{7}$$

The pixel positions of $\boldsymbol{f}$ after displacement and deformation, $\boldsymbol{x}_i' = [x_i' \quad y_i']^T$, (referred to as query points) are determined as

$$\boldsymbol{x}_i' = \Delta \boldsymbol{x}_i' + \boldsymbol{x}^o. \tag{8}$$

Thus $\Delta \boldsymbol{x}_i'$ represents the distance from the center of $\boldsymbol{f}$ to $\boldsymbol{x}_i'$. The light intensity values of G are interpolated at $\boldsymbol{x}_i'$ to obtain $\boldsymbol{g}$. Thus, $\boldsymbol{g}$ represents a portion of G at a location described by the displacement SFPs with the deformation defined by the SFPs having been removed. The correlation criterion quantifies the degree of similarity between $\boldsymbol{f}$ and $\boldsymbol{g}$.

2.3.1. Shape Function

The main SFs are the zero ($\boldsymbol{W}^{SF0}$), first ($\boldsymbol{W}^{SF1}$), and second-order SFs ($\boldsymbol{W}^{SF2}$) given as [43]

$$\begin{gathered} \boldsymbol{W}^{SF0}(\Delta \boldsymbol{x}_i, \boldsymbol{P}^{SF0}) = \begin{bmatrix} 1 & 0 & u \\ 0 & 1 & v \end{bmatrix} \begin{bmatrix} \Delta x_i \\ \Delta y_i \\ 1 \end{bmatrix}, \\ \boldsymbol{W}^{SF1}(\Delta \boldsymbol{x}_i, \boldsymbol{P}^{SF1}) = \begin{bmatrix} 1+u_x & u_y & u \\ v_x & 1+v_y & v \end{bmatrix} \begin{bmatrix} \Delta x_i \\ \Delta y_i \\ 1 \end{bmatrix} \\ \text{and } \boldsymbol{W}^{SF2}(\Delta \boldsymbol{x}_i, \boldsymbol{P}^{SF2}) = \begin{bmatrix} \frac{1}{2}u_{xx} & u_{xy} & \frac{1}{2}u_{yy} & 1+u_x & u_y & u \\ \frac{1}{2}v_{xx} & v_{xy} & \frac{1}{2}v_{yy} & v_x & 1+v_y & v \end{bmatrix} \begin{bmatrix} \Delta x_i^2 \\ \Delta x_i \Delta y_i \\ \Delta y_i^2 \\ \Delta x_i \\ \Delta y_i \\ 1 \end{bmatrix}. \end{gathered} \tag{9}$$

Their sets of SFPs are given as

$$\begin{gathered} \boldsymbol{P}^{SF0} = [u \quad v]^T, \\ \boldsymbol{P}^{SF1} = [u \quad u_x \quad u_y \quad v \quad v_x \quad v_y]^T \\ \text{and } \boldsymbol{P}^{SF2} = [u \quad u_x \quad u_y \quad u_{xx} \quad u_{xy} \quad u_{yy} \quad v \quad v_x \quad v_y \quad v_{xx} \quad v_{xy} \quad v_{yy}]^T. \end{gathered} \tag{10}$$

Here u and v represent the displacement of $\boldsymbol{f}$ in the x- and y-directions respectively. Furthermore, u_x, u_{xx}, v_y, and v_{yy} define stretching of $\boldsymbol{f}$, whereas u_y, v_x, u_{yy}, v_{xx}, u_{xy}, and v_{xy} define shearing. Although higher-order SFs more reliably determine displacements in complex displacement fields by accounting for more complex deformation, the DIC process is only concerned with the displacements determined by correlation.

2.3.2. Interpolation

ADIC3D employs the bi-cubic b-spline interpolation method detailed in the work of Hou et al. [63]. Letting F and G represent interpolation functions, the light intensity values of pixel i of the reference, f_i, and investigated subsets, g_i, are interpolated as

$$f_i = F(\boldsymbol{x}^o + \Delta \boldsymbol{x}_i) \text{ and } g_i = G(\boldsymbol{x}^o + \boldsymbol{W}(\Delta \boldsymbol{x}_i, \boldsymbol{P})). \tag{11}$$

Images are filtered using a Gaussian low-pass filter, detailed by Pan [45], since otherwise the sensitivity of bi-cubic b-spline interpolation to high frequency noise would lead to bias in the displacements. The Gaussian filtering parameters, being the window size (β) and the standard deviation of the Gaussian function (σ^g), should be chosen to reduce bias without increasing variance.

2.3.3. Correlation Criterion

The objective function employs the more computationally efficient ZNSSD correlation criterion (C_{ZNSSD}), having a range of $\{C_{ZNSSD} \in \mathbb{R} | 0 \leq C_{ZNSSD} \leq 4\}$ where smaller values indicate higher similarity, given as

$$C_{ZNSSD} = \sum_{i=1}^{I} \left[\frac{f_i - \bar{f}}{\tilde{f}} - \frac{g_i - \bar{g}}{\tilde{g}} \right]^2, \tag{12}$$

where

$$\bar{f} = \frac{\sum_i^I f_i}{I},\ \bar{g} = \frac{\sum_i^I g_i}{I}, \tag{13}$$

$$\tilde{f} = \sqrt{\sum_{i=1}^{I} (f_i - \bar{f})^2} \text{ and } \tilde{g} = \sqrt{\sum_{i=1}^{I} (g_i - \bar{g})^2}. \tag{14}$$

Here there are I many pixels per subset. The correlation coefficient is reported as the zero-mean normalized cross correlation criterion (ZNCC), C_{ZNCC}, since its range, $\{C_{ZNCC} \in \mathbb{R} | -1 \leq C_{ZNCC} \leq 1\}$ where larger values indicate higher similarity, is more intuitive. These correlation criteria are related as [64]

$$C_{ZNCC} = 1 - \frac{C_{ZNSSD}}{2}. \tag{15}$$

2.3.4. Objective Function

The objective function, based on Equation (12), is given as

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

Therefore, correlation seeks the SFPs that define $\boldsymbol{g}$ to contain a light intensity pattern similar to $\boldsymbol{f}$ thereby optimizing C_{ZNSSD}. However, the IC-GN optimization method requires modifying the objective function in order to derive an iterative optimization equation from it. More specifically, the current estimate of the SFPs, $\boldsymbol{P}$, is applied to the investigated subset while the iterative improvement of the SFPs, $\Delta\boldsymbol{P}$, is applied to the reference subset as

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

2.3.5. Optimization Equation

The optimization equation derived from Equation (17), derivation detailed in [53], is given as

$$\Delta\boldsymbol{P} = -\boldsymbol{H}^{-1} \sum_{i=1}^{I} \left(\nabla \boldsymbol{f}_i \frac{\partial \boldsymbol{W}_i}{\partial \boldsymbol{P}} \right)^T \left[f_i - \bar{f} - \frac{\tilde{f}}{\tilde{g}} \left(G(\boldsymbol{x}^o + \boldsymbol{W}(\Delta \boldsymbol{x}_i, \boldsymbol{P})) - \bar{g} \right) \right]. \tag{18}$$

Here $\boldsymbol{H}$ is the Hessian, defined in Equation (19), and the remaining terms (within the summation) form the Jacobian, $\boldsymbol{J}$. $\boldsymbol{H}$, being independent of $\boldsymbol{P}$, is precomputed prior to beginning iterations.

$$\boldsymbol{H} = \sum_{i=1}^{I} \left[\left(\nabla \boldsymbol{f}_i \frac{\partial \boldsymbol{W}_i}{\partial \boldsymbol{P}} \right)^T \left(\nabla \boldsymbol{f}_i \frac{\partial \boldsymbol{W}_i}{\partial \boldsymbol{P}} \right) \right]. \tag{19}$$

$\nabla \boldsymbol{f}_i = \begin{bmatrix} \frac{\partial f_i}{\partial x} & \frac{\partial f_i}{\partial y} \end{bmatrix}$ is the light intensity gradient of $\boldsymbol{f}$ at pixel i. $\frac{\partial \boldsymbol{W}_i}{\partial \boldsymbol{P}}$ is the Jacobian of the SF for pixel i and based on the SF order is given as

$$\frac{\partial \boldsymbol{W}_i^{SF0}}{\partial \boldsymbol{P}^{SF0}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$
$$\frac{\partial \boldsymbol{W}_i^{SF1}}{\partial \boldsymbol{P}^{SF1}} = \begin{bmatrix} 1 & \Delta x_i & \Delta y_i & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta x_i & \Delta y_i \end{bmatrix} \tag{20}$$

and $\frac{\partial \boldsymbol{W}_i^{SF2}}{\partial \boldsymbol{P}^{SF2}} = \begin{bmatrix} 1 & \Delta x_i & \Delta y_i & \frac{\Delta x_i^2}{2} & \Delta x_i \Delta y_i & \frac{\Delta y_i^2}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta x_i & \Delta y_i & \frac{\Delta x_i^2}{2} & \Delta x_i \Delta y_i & \frac{\Delta y_i^2}{2} \end{bmatrix}$.

Although, each iteration of Equation (18) attempts to determine how $\boldsymbol{f}$ should displace and deform (according to $\Delta \boldsymbol{P}$) such that it matches $\boldsymbol{g}$, $\boldsymbol{f}$ is not displaced or deformed in practice. Instead $\Delta \boldsymbol{P}$ is used to determine the updated SFPs, $\boldsymbol{P}_{update}$, which serves as the current estimate in the next iteration. The stopping criterion deems when further iterations are redundant.

2.3.6. Updating the SFPs

$\boldsymbol{P}_{update}$ is obtained by composing the inverse of $\Delta \boldsymbol{P}$ with $\boldsymbol{P}$ as

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

Here $\boldsymbol{\omega}$, being dependent on the SF order, uses the SFPs to populate a square matrix as [43]

$$\boldsymbol{\omega}^{SF0}(\boldsymbol{P}^{SF0}) = \begin{bmatrix} 1 & 0 & u \\ 0 & 1 & v \\ 0 & 0 & 1 \end{bmatrix},$$

$$\boldsymbol{\omega}^{SF1}(\boldsymbol{P}^{SF1}) = \begin{bmatrix} 1+u_x & u_y & u \\ v_x & 1+v_y & v \\ 0 & 0 & 1 \end{bmatrix}$$

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

where

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

2.3.7. Stopping Criterion

The stopping criterion, $||\Delta \boldsymbol{P}||$, determines the magnitude of the change in the SFPs between iterations, based on $\Delta \boldsymbol{P}$, as [49]

$$||\Delta \boldsymbol{P}^{SF0}|| = [\Delta u^2 + \Delta v^2]^{0.5},$$

$$||\Delta \boldsymbol{P}^{SF1}|| = \left[\Delta u^2 + (\Delta u_x \zeta)^2 + \left(\Delta u_y \zeta\right)^2 + \Delta v^2 + (\Delta v_x \zeta)^2 + \left(\Delta v_y \zeta\right)^2\right]^{0.5}$$

$$\text{and } ||\Delta \boldsymbol{P}^{SF2}|| = \Big[\Delta u^2 + (\Delta u_x \zeta)^2 + (\Delta u_y \zeta)^2 + (\tfrac{1}{2}\Delta u_{xx}\zeta^2)^2 + (\Delta u_{xy}\zeta^2)^2 + (\tfrac{1}{2}\Delta u_{yy}\zeta^2)^2 + \Delta v^2 + (\Delta v_x \zeta)^2 + (\Delta v_y \zeta)^2 + (\tfrac{1}{2}\Delta v_{xx}\zeta^2)^2 + (\Delta v_{xy}\zeta^2)^2 + (\tfrac{1}{2}\Delta v_{yy}\zeta^2)^2\Big]^{0.5}. \quad (23)$$

Here ζ is the largest value of Δx_i or Δy_i for the subset. Iterations cease once $||\Delta \boldsymbol{P}||$ falls below a predefined value called the stopping criterion value.

2.4. Epipolar Geometry

Epipolar geometry is the projective geometry between the ideal sensor CSs, i.e., between camera 1 and 2. Consider the following definitions aided by Figure 2: (i) the baseline is formed between the two origins of the camera CSs; (ii); the epipolar plane is formed between the 3D coordinate and the baseline; (iii) the epipolar lines, $\underline{\boldsymbol{\lambda}}_j$, are the intersection of the epipolar plane and the x-y plane of the ideal sensor CSs; and (iv) the epipoles, given as $\underline{\boldsymbol{e}}_{s_j} = \begin{bmatrix} e^1_{s_j} & e^2_{s_j} & e^3_{s_j} \end{bmatrix}^T$ in homogeneous coordinates, are located at the intersection of the baseline and the x–y plane of the ideal sensor CSs.

For 2D coordinates in the ideal sensor CSs of the first, $\underline{\hat{\boldsymbol{x}}}_{s_1}$, and second cameras, $\underline{\hat{\boldsymbol{x}}}_{s_2}$, to correspond to the same $\hat{\boldsymbol{x}}_w$ requires that these coordinates satisfy the epipolar constraint defined as

$$\underline{\hat{\boldsymbol{x}}}_{s_2}{}^T \boldsymbol{B} \underline{\hat{\boldsymbol{x}}}_{s_1} = 0. \tag{24}$$

More specifically, since the fundamental matrix, $\boldsymbol{B}$, maps a point in one ideal sensor CS to its corresponding epipolar line in the other ideal sensor CS, the 2D coordinates must lie on the epipolar lines of their respective ideal sensor CSs. $\boldsymbol{B}$ encapsulates the intrinsic and extrinsic relation between the ideal sensor CSs.

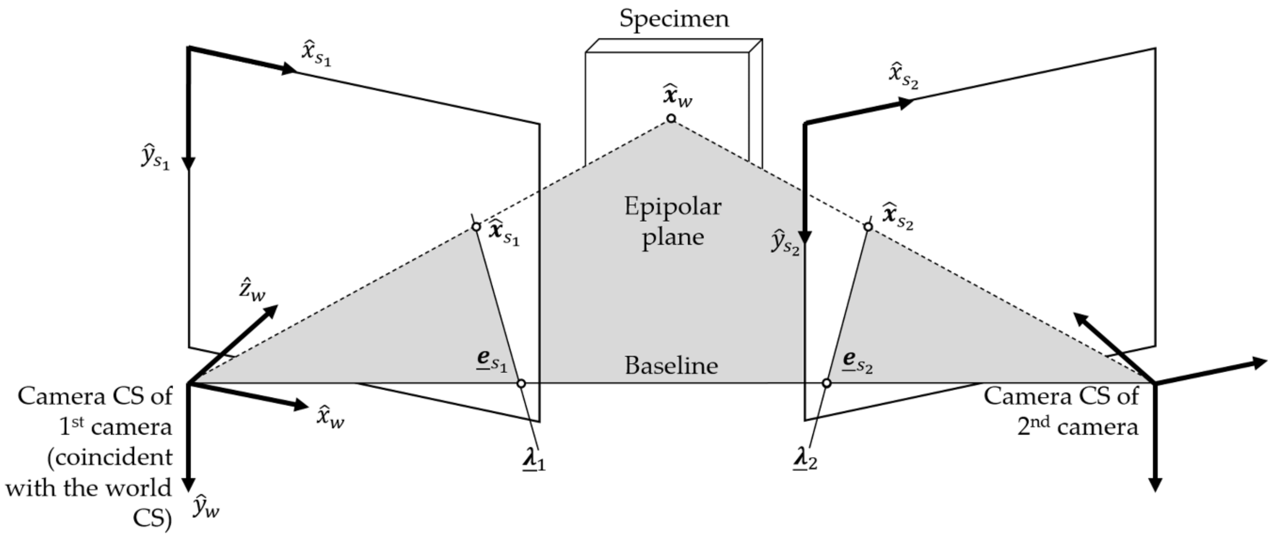
Specimen
$\hat{x}_{s_1}$
$\hat{x}_{s_2}$
$\hat{\boldsymbol{x}}_w$
$\hat{y}_{s_1}$
Epipolar plane
$\hat{\boldsymbol{x}}_{s_1}$
$\hat{\boldsymbol{x}}_{s_2}$
$\hat{y}_{s_2}$
$\hat{z}_w$
$\underline{\boldsymbol{e}}_{s_1}$
Baseline
$\underline{\boldsymbol{e}}_{s_2}$
Camera CS of 1st camera (coincident with the world CS)
$\hat{x}_w$
$\hat{y}_w$
$\underline{\boldsymbol{\lambda}}_1$
$\underline{\boldsymbol{\lambda}}_2$
Camera CS of 2nd camera

Figure 2. Schematic illustration of the epipolar geometry.

Computing $\boldsymbol{B}$ requires applying a homography, $\boldsymbol{\Omega}$, to the projection matrices of the cameras to transform them to canonical form ($\boldsymbol{Q}_{c_j}$), where the world CS coincides with the camera CS of the first camera, as [65]

$$\boldsymbol{Q}_{c_1} = \boldsymbol{Q}_1 \boldsymbol{\Omega} = [\boldsymbol{I}|\boldsymbol{0}] \text{ and } \boldsymbol{Q}_{c_2} = \boldsymbol{Q}_2 \boldsymbol{\Omega} = [\boldsymbol{M}|\boldsymbol{m}] \tag{25}$$

where

$$\boldsymbol{\Omega} = \begin{bmatrix} Q_1^{11} & Q_1^{12} & Q_1^{13} & Q_1^{14} \\ Q_1^{21} & Q_1^{22} & Q_1^{23} & Q_1^{24} \\ Q_1^{31} & Q_1^{32} & Q_1^{33} & Q_1^{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}. \tag{26}$$

Here $\boldsymbol{I}$ is a 3×3 identity matrix. $\boldsymbol{Q}_{c_2}$, which consists of a 3×3 matrix $\boldsymbol{M}$ and a 3×1 vector $\boldsymbol{m}$, transforms a coordinate from the camera CS of the first camera to its location in the ideal sensor CS of the second camera. $\boldsymbol{B}$ is given as

$$\boldsymbol{B} = [\boldsymbol{m}]_\times \boldsymbol{M}, \tag{27}$$

where $[\boldsymbol{m}]_\times$ is the skew-symmetric matrix of $\boldsymbol{m} = [m^1 \quad m^2 \quad m^3]^T$ defined as

$$[\boldsymbol{m}]_\times = \begin{bmatrix} 0 & -m^3 & m^2 \\ m^3 & 0 & -m^1 \\ -m^2 & m^1 & 0 \end{bmatrix}. \tag{28}$$

2.5. Stereo-DIC Overview

The core operations of Stereo-DIC are subset matching, calibration, and displacement transformation. For a point on the specimen, subset matching determines the two sets of in-plane, pixel displacements it experiences in each distorted sensor CS which, along with the calibration parameters determined by the calibration process, are used by displacement transformation to determine the in-plane and out-of-plane (thus 3D) metric displacements it experiences in the world CS. Note that with regards to subset matching and displacement transformation, the term *in-plane* refers to the x–y plane of the distorted sensor CSs and world CS respectively.

Subset matching is composed of stereo and temporal matching which utilize correlation differently. First, subset matching breaks up the first image of the first image series (FIS1) into subsets. Second, stereo matching determines the location of the corresponding subsets in the first image of the second image series (FIS2). Such subsets, one from each image of an image pair which correspond to the same point on the specimen, are termed a subset pair. Lastly, temporal matching, processing each image series separately, determines the displacements experienced by these subsets through their respective image series. This is achieved by sequentially stepping through an image series, starting at the second image, and at each step performing correlation by treating the current image as the deformed image while the reference image is dictated by the reference image strategy as discussed in Section 2.6.1. Thus, stereo matching is unique to Stereo-DIC while temporal matching is identical to that of 2D DIC.

The calibration method is identical to that of 2D DIC. However, in order to determine the epipolar geometry from the calibration parameters requires that the two calibration image series, used to calibrate the cameras separately, form part of the same calibration image set.

Displacement transformation, illustrated in Figure 3, steps through image pairs of the image set similarly to temporal matching. More specifically, considering a single reference subset pair with known locations, $\boldsymbol{x}_{s_j}^o$, it uses the in-plane displacements determined by temporal matching (u_j and v_j in the x- and y-directions) between the reference images (of the reference image pair) and the deformed images (of the deformed image pair) to determine the locations of the investigated subset pair, $\boldsymbol{x}_{s_j}^d$.

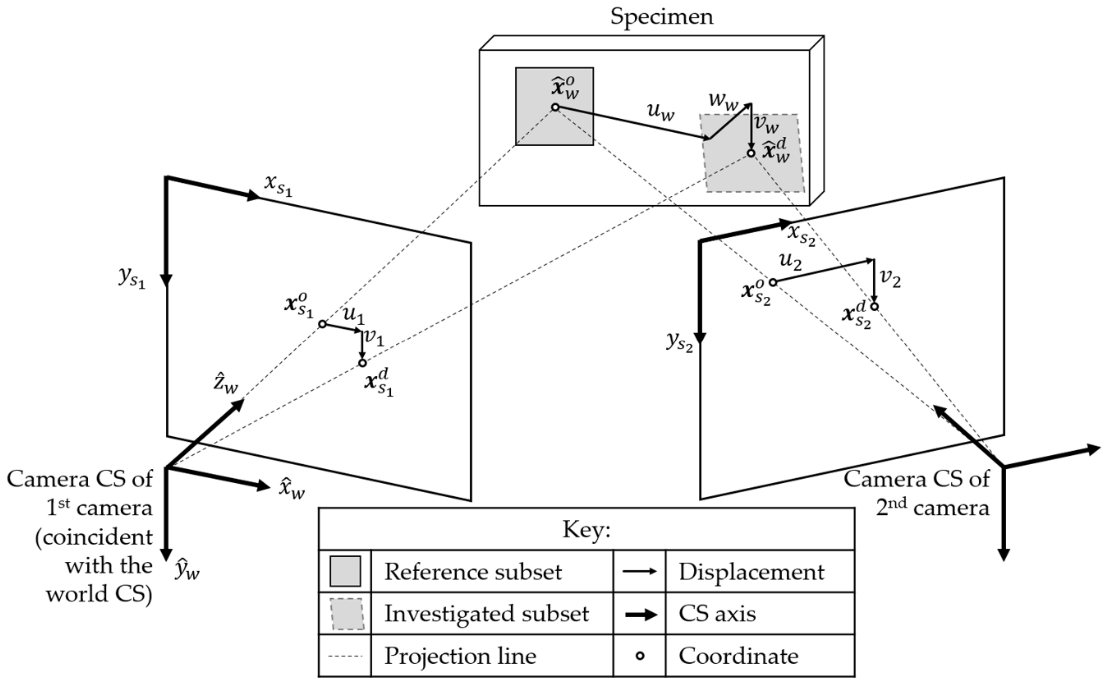

Figure 3. Schematic illustration of the displacement transformation process. Note that a projection line represents the combined process of distortion correction and triangulation.

In contrast to 2D DIC, distortion correction alone is insufficient to determine the ideal coordinate of a subset in the ideal sensor CS. More specifically, discretization of light to form an image introduces noise in the images [66] and displacements measured by correlation, which are used to determine the locations of subsets, inherently contain some noise. Thus, correcting $\boldsymbol{x}_{s_j}^o$ and $\boldsymbol{x}_{s_j}^d$ for radial distortion results in the measured coordinates of the reference, $\breve{\boldsymbol{x}}_{s_j}^o$, and investigated subset pair, $\breve{\boldsymbol{x}}_{s_j}^d$, in the ideal sensor CS respectively. Generally, these measured coordinates of a subset pair, represented as $\breve{\boldsymbol{x}}_{s_j}$, do not satisfy the epipolar constraint of Equation (24) since their back-projection rays do not intersect. Thus, triangulation, which determines the 3D coordinate in the world CS, $\hat{\boldsymbol{x}}_w$, from $\breve{\boldsymbol{x}}_{s_j}$, is performed in two steps. First the polynomial triangulation method, detailed in Section 2.7, determines the ideal coordinates of a subset pair, $\hat{\boldsymbol{x}}_{s_j}$, which lie as close to $\breve{\boldsymbol{x}}_{s_j}$ as possible while satisfying the epipolar constraint. Thereafter the linear triangulation method, detailed in Section 2.8, determines $\hat{\boldsymbol{x}}_w$ to which $\hat{\boldsymbol{x}}_{s_j}$ correspond as the intersection of their back-projected rays. Performing triangulation on $\breve{\boldsymbol{x}}_{s_j}^o$ and $\breve{\boldsymbol{x}}_{s_j}^d$ results in $\hat{\boldsymbol{x}}_w^o$ and $\hat{\boldsymbol{x}}_w^d$ respectively.

Finally, $\hat{\boldsymbol{x}}_w^o$ is subtracted from $\hat{\boldsymbol{x}}_w^d$ to determine the displacement experienced by the point on the specimen, in the x-, y-, and z-directions represented as $\hat{u}_w$, $\hat{v}_w$, and $\hat{w}_w$ respectively, between the time stamps at which the reference and deformed image pairs were captured. Note that subscript w indicates coordinates or displacements in the world CS, whereas variable w indicates displacement in the z-direction. This is done for all subset pairs across all image pair combinations, analyzed by temporal matching, to determine the full-field metric displacements experienced by the specimen throughout the image set.

2.6. Subset Matching

The optical flow determined by temporal and stereo matching is a result of the displacement and/or deformation experienced by the specimen (over time) and the perspective change between the cameras respectively. They differ predominantly in the method used to obtain SFP initial estimates.

2.6.1. Temporal Matching

There are two reference image strategies for temporal matching: absolute and incremental. The absolute and incremental strategies define the reference image as either the first image of the image series or the previous image, relative to the current deformed image, respectively. The incremental strategy more reliably tracks more severe deformations between images but incurs accumulative errors when determining the displacement relative to the first image.

Temporal matching uses the Phase Correlation Method (PCM) to determine initial estimates for the displacements between two subsets. PCM efficiently computes the correlation coefficients in the frequency domain for a range of integer displacements of up to half the subset size in each direction between the two subsets. The integer displacements with the best correlation coefficient are identified and used as initial estimates for the displacement SFPs while the deformation SFPs are set to zero. Consult the work of Foroosh et al. [67] for a discussion on PCM. For the absolute strategy, PCM is only used for the first correlation run. Thereafter the SFPs of the previous correlation run, of the same image series, are used as an initial estimate for the SFPs of the current correlation run. The incremental strategy uses PCM for each correlation run.

2.6.2. Stereo Matching

Determining reliable out-of-plane displacements in the world CS requires an angle between the optical axes of the cameras of between 15° and 35° [5,68]. As such, stereo matching uses the second-order SF since subsets experience complex deformation between the FIS1 and FIS2 due to this perspective change [41].

Traditionally, template matching, which utilizes image rectification, was used to determine SFP initial estimates for stereo matching [69]. However, image rectification is susceptible to errors in the calibration parameters, can cause aliasing in the rectified images and interpolation during image rectification can lead to errors in the determined SFPs [70].

As such, feature matching methods are becoming favored [41]. ADIC3D's feature matching method is based on that proposed by Zhou et al. [71]. It uses the scale-invariant feature transform (SIFT) feature matching algorithm [72] to identify corresponding keypoints between the images (where a keypoint is a unique feature in an image) and the affine transformation mapping model to mathematically relate the coordinates of these keypoint pairs between the images.

The SIFT algorithm is used because it is robust in translation, rotation, image scaling, variation in illumination, affine transformation and moderate perspective change. SIFT identifies keypoints that are scale invariant and computes a descriptor, in the form of a 128-dimensional vector, for each keypoint based on the light intensity gradient information in the vicinity of the keypoint.

For each keypoint of the FIS1, the Euclidean distance between its descriptor and that of every keypoint of the FIS2 is determined. The two keypoints of the FIS2 with the smallest descriptor distances are identified. If the ratio of the smallest to second smallest descriptor distance is less than 0.8, the keypoint of the FIS2 with the smallest descriptor distance is designated as the matching keypoint of the keypoint of the FIS1 [72]. Otherwise, this keypoint of the FIS1 is discarded.

The affine transformation relates a keypoint pair as

$$\begin{bmatrix} x_{s_2}^k \\ y_{s_2}^k \end{bmatrix} = \begin{bmatrix} 1+a_1 & a_2 & a_3 \\ a_4 & 1+a_5 & a_6 \end{bmatrix} \begin{bmatrix} x_{s_1}^k \\ y_{s_1}^k \\ 1 \end{bmatrix}. \tag{29}$$

Here $\boldsymbol{x}_{s_1}^k = \begin{bmatrix} x_{s_1}^k & y_{s_1}^k \end{bmatrix}^T$ and $\boldsymbol{x}_{s_2}^k = \begin{bmatrix} x_{s_2}^k & y_{s_2}^k \end{bmatrix}^T$ are the locations of the kth keypoint in the FIS1 and FIS2 respectively while $a_1$through a_6 are the affine transformation parameters. Given three or more keypoint pairs, the affine transformation parameters can be solved either as a system of linear equations or through linear least-squares respectively.

The K many $\boldsymbol{x}_{s_1}^k$ that are nearest to the subset's center position are used to determine the affine transformation parameters of the subset. However, since SIFT often returns false keypoint pairs, the m-estimator sample consensus (MSAC) method [73] is used to remove these false keypoint pairs when determining the affine transformation parameters. False keypoint pairs are identified as those which have a squared error distance greater than a predefined threshold, τ. The squared error distance of the kth keypoint pair, E_k^{dist}, is calculated as

$$E_k^{dist} = \left(x_{s_2}^k - (1 + a_1)x_{s_1}^k - a_2 y_{s_1}^k - a_3\right)^2 + \left(y_{s_2}^k - a_4 x_{s_1}^k - (1 + a_5)y_{s_1}^k - a_6\right)^2. \quad (30)$$

ADIC3D uses $K = 20$, $\tau = 1$ pixel^2 and a confidence, specifying the probability that MSAC finds the maximum number of inliers, of 99.5%. Second-order SFP initial estimates are determined from the affine transformation parameters and the subset position in the FIS1, $\boldsymbol{x}_{s_1} = [x_{s_1} \quad y_{s_1}]^T$, as

$$\begin{aligned} u &= a_1 x_{s_1} + a_2 y_{s_1} + a_3, & v &= a_4 x_{s_1} + a_5 y_{s_1} + a_6, \\ u_x &= a_1, & v_x &= a_4, \\ u_y &= a_2, & v_y &= a_5, \\ u_{xx} &= 0, & v_{xx} &= 0, \\ u_{xy} &= 0, & v_{xy} &= 0, \\ u_{yy} &= 0 & \text{and } v_{yy} &= 0. \end{aligned} \quad (31)$$

Once correlation has optimized these SFPs, the corresponding subset position in the FIS2, $\boldsymbol{x}_{s_2} = [x_{s_2} \quad y_{s_2}]^T$, is determined as

$$\begin{bmatrix} x_{s_2} \\ y_{s_2} \end{bmatrix} = \begin{bmatrix} x_{s_1} \\ y_{s_1} \end{bmatrix} + \begin{bmatrix} u \\ v \end{bmatrix}. \quad (32)$$

2.7. Polynomial Triangulation Method

The polynomial triangulation method, proposed by Hartley and Sturm [66], determines $\hat{\boldsymbol{x}}_{s_j}$ by minimizing the geometric error cost function, D, given as

$$D = \delta\left(\check{\boldsymbol{x}}_{s_1}, \hat{\boldsymbol{x}}_{s_1}\right)^2 + \delta\left(\check{\boldsymbol{x}}_{s_2}, \hat{\boldsymbol{x}}_{s_2}\right)^2 \text{ subject to } \underline{\hat{\boldsymbol{x}}}_{s_2}{}^T \boldsymbol{B} \underline{\hat{\boldsymbol{x}}}_{s_1} = 0. \quad (33)$$

Here function δ determines the Euclidean distance between two points. Determining $\hat{\boldsymbol{x}}_{s_j}$ by minimizing the distance between the coordinates $\hat{\boldsymbol{x}}_{s_j}$ and $\check{\boldsymbol{x}}_{s_j}$ within the ideal sensor CSs ensures that the polynomial triangulation method is projective-invariant. Projective-invariance implies that the projective frame within which $\hat{\boldsymbol{x}}_w$ is defined does not affect the values of $\hat{\boldsymbol{x}}_{s_j}$ [65]. The polynomial method assumes that the calibration parameters are known with greater accuracy than the measured coordinates [66]. Furthermore, it is assumed that neither $\hat{\boldsymbol{x}}_{s_j}$ nor $\check{\boldsymbol{x}}_{s_j}$ coincides with an epipole since this would lead to an unfavorable solution for the 3D coordinate.

It is known that a pair of ideal coordinates which satisfy the epipolar constraint lie on a pair of corresponding epipolar lines. Additionally, the epipolar plane can only rotate about the baseline since the camera CSs, and thus epipoles, are fixed. Thus, there is a family of epipolar lines upon which the idea coordinates can lie. Furthermore, the projective-invariance of the polynomial triangulation method allows applying a separate rigid transformation to each ideal sensor CS in order to parameterize the family of epipolar lines in terms of a single variable t. As such, homogeneous coordinates are utilized and Equation (32) becomes

$$D = \delta\left(\underline{\check{\boldsymbol{x}}}_{s_1}, \underline{\boldsymbol{\lambda}}_1(t)\right)^2 + \delta\left(\underline{\check{\boldsymbol{x}}}_{s_2}, \underline{\boldsymbol{\lambda}}_2(t)\right)^2, \quad (34)$$

where $\underline{\boldsymbol{\lambda}}_j(t)$ is a vector representing an epipolar line in the ideal sensor CS as a function of t while $\delta\left(\underline{\check{\boldsymbol{x}}}_{s_j}, \underline{\boldsymbol{\lambda}}_j(t)\right)$ represents the distance from $\underline{\check{\boldsymbol{x}}}_{s_j}$ to its orthogonal projection on

$\underline{\boldsymbol{\lambda}}_j(t)$. The rigid transformations first apply a translation matrix, $\boldsymbol{T}_j^{pm}$, to each ideal sensor CS translating $\underline{\check{\boldsymbol{x}}}_{s_j}$ to the origin of its CS.

$$\boldsymbol{T}_1^{pm} = \begin{bmatrix} 1 & 0 & -\check{x}_{s_1} \\ 0 & 1 & -\check{y}_{s_1} \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \boldsymbol{T}_2^{pm} = \begin{bmatrix} 1 & 0 & -\check{x}_{s_2} \\ 0 & 1 & -\check{y}_{s_2} \\ 0 & 0 & 1 \end{bmatrix}. \quad (35)$$

The original fundamental matrix, $\boldsymbol{B}$, determined using Equation (27) needs to be updated as

$$\boldsymbol{B}_1 = \boldsymbol{T}_2^{pm^{-T}} \boldsymbol{B} \boldsymbol{T}_1^{pm^{-1}}. \quad (36)$$

The epipoles $\underline{\boldsymbol{e}}_{s_1}$ and $\underline{\boldsymbol{e}}_{s_2}$ of the first and second cameras are computed, using singular value decomposition, as the right and left null-space of $\boldsymbol{B}_1$ respectively. Thereafter they are normalized such that $\left(e_{s_1}^1\right)^2 + \left(e_{s_1}^2\right)^2 = 1$ and $\left(e_{s_2}^1\right)^2 + \left(e_{s_2}^2\right)^2 = 1$.

The second step of the rigid transformation applies rotations $\boldsymbol{R}_1^{pm}$ and $\boldsymbol{R}_2^{pm}$ to the ideal sensor CSs of the first and second cameras respectively in order to place their epipoles on the x-axes of their respective CSs as $\boldsymbol{R}_1^{pm}\underline{\boldsymbol{e}}_{s_1} = \begin{bmatrix} 1 & 0 & e_{s_1}^3 \end{bmatrix}^T$ and $\boldsymbol{R}_2^{pm}\underline{\boldsymbol{e}}_{s_2} = \begin{bmatrix} 1 & 0 & e_{s_2}^3 \end{bmatrix}^T$.

$$\boldsymbol{R}_1^{pm} = \begin{bmatrix} e_{s_1}^1 & e_{s_1}^2 & 0 \\ -e_{s_1}^2 & e_{s_1}^1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \boldsymbol{R}_2^{pm} = \begin{bmatrix} e_{s_2}^1 & e_{s_2}^2 & 0 \\ -e_{s_2}^2 & e_{s_2}^1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (37)$$

The fundamental matrix is updated as

$$\boldsymbol{B}_2 = \boldsymbol{R}_2^{pm} \boldsymbol{B}_1 \boldsymbol{R}_1^{pm^T}. \quad (38)$$

Since $\boldsymbol{B}_2\begin{bmatrix} 1 & 0 & e_{s_1}^3 \end{bmatrix}^T = \begin{bmatrix} 1 & 0 & e_{s_2}^3 \end{bmatrix}\boldsymbol{B}_2 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$ the elements of the fundamental matrix are represented as

$$\boldsymbol{B}_2 = \begin{bmatrix} e_{s_1}^3 e_{s_2}^3 \varphi_4 & -e_{s_2}^3 \varphi_3 & -e_{s_2}^3 \varphi_4 \\ -e_{s_1}^3 \varphi_2 & \varphi_1 & \varphi_2 \\ -e_{s_1}^3 \varphi_4 & \varphi_3 & \varphi_4 \end{bmatrix}. \quad (39)$$

An epipolar line of the first camera, $\underline{\boldsymbol{\lambda}}_1(t)$, passing through point $\begin{bmatrix} 0 & t & 1 \end{bmatrix}^T$ on the y-axis of the ideal sensor CS is represented as

$$\underline{\boldsymbol{\lambda}}_1(t) = \begin{bmatrix} 0 & t & 1 \end{bmatrix}^T \times \begin{bmatrix} 1 & 0 & e_{s_1}^3 \end{bmatrix}^T = \begin{bmatrix} te_{s_1}^3 & 1 & -t \end{bmatrix}^T. \quad (40)$$

The corresponding epipolar line of the second camera, $\underline{\boldsymbol{\lambda}}_2(t)$, is computed as

$$\underline{\boldsymbol{\lambda}}_2(t) = \boldsymbol{B}_2\begin{bmatrix} 0 & t & 1 \end{bmatrix}^T = \begin{bmatrix} -e_{s_2}^3(\varphi_3 t + \varphi_4) & \varphi_1 t + \varphi_2 & \varphi_3 t + \varphi_4 \end{bmatrix}^T. \quad (41)$$

Thus, $\delta\left(\underline{\check{\boldsymbol{x}}}_{s_1}, \underline{\boldsymbol{\lambda}}_1(t)\right)^2 = \frac{t^2}{1+\left(te_{s_1}^3\right)^2}$ and $\delta\left(\underline{\check{\boldsymbol{x}}}_{s_2}, \underline{\boldsymbol{\lambda}}_2(t)\right)^2 = \frac{(\varphi_3 t+\varphi_4)^2}{(\varphi_1 t+\varphi_2)^2 + {e_{s_2}^3}^2(\varphi_3 t+\varphi_4)^2}$ and so Equation (34) has the form

$$D(t) = \frac{t^2}{1+\left(te_{s_1}^3\right)^2} + \frac{(\varphi_3 t+\varphi_4)^2}{(\varphi_1 t+\varphi_2)^2 + {e_{s_2}^3}^2(\varphi_3 t+\varphi_4)^2}. \quad (42)$$

Finding the optimal value of t involves taking the derivative Equation (42) in terms of t, collecting terms over a common denominator and equating the resulting numerator to 0 as

$$t\left((\varphi_1 t + \varphi_2)^2 + {e_{s_2}^3}^2(\varphi_3 t + \varphi_4)^2\right)^2 - (\varphi_1\varphi_4 - \varphi_2\varphi_3)\left(1 + \left(te_{s_1}^3\right)^2\right)^2 (\varphi_1 t + \varphi_2)(\varphi_3 t + \varphi_4) = 0. \quad (43)$$

The 6 roots of Equation (43) are solved for and Equation (42) is evaluated at the real parts of these roots to determine the value of t which corresponds to a minimum. The asymptotic value of Equation (42) should also be evaluated as $t \rightarrow \infty$ which corresponds to an epipolar line in the first image of $e_{s_1}^3 \breve{x}_{s_1} = 1$ [65].

The optimal t is used to determine $\hat{\boldsymbol{x}}_{s_1}$ and $\hat{\boldsymbol{x}}_{s_2}$ as

$$\hat{\boldsymbol{x}}_{s_1} = \Phi\left(\boldsymbol{T}_1^{pm^{-1}} \boldsymbol{R}_1^{pm^T} \begin{bmatrix} t^2 e_{s_1}^3 \\ t \\ \left(t e_{s_1}^3\right)^2 + 1 \end{bmatrix}\right) \tag{44}$$

$$\text{and } \hat{\boldsymbol{x}}_{s_2} = \Phi\left(\boldsymbol{T}_2^{pm^{-1}} \boldsymbol{R}_2^{pm^T} \begin{bmatrix} e_{s_2}^3 (\varphi_3 t + \varphi_4)^2 \\ -(\varphi_1 t + \varphi_2)(\varphi_3 t + \varphi_4) \\ (\varphi_1 t + \varphi_2)^2 + {e_{s_2}^3}^2 (\varphi_3 t + \varphi_4)^2 \end{bmatrix}\right). \tag{45}$$

Note that the inverse of the rigid transformations are applied during Equations (44) and (45) such that $\hat{\boldsymbol{x}}_{s_1}$ and $\hat{\boldsymbol{x}}_{s_2}$ are in the original ideal sensor CSs.

2.8. Linear Triangulation Method

Denoting the nth row of $\boldsymbol{Q}_{c_j}$ as $\boldsymbol{q}_{c_j}^n$ the projection of $\hat{\boldsymbol{x}}_w$ to $\hat{\boldsymbol{x}}_{s_1}$ is described by the following three equations

$$\alpha_1 \hat{x}_{s_1} = {\boldsymbol{q}_{c_1}^1}^T \hat{\boldsymbol{x}}_w,\ \alpha_1 \hat{y}_{s_1} = {\boldsymbol{q}_{c_1}^2}^T \hat{\boldsymbol{x}}_w \text{ and } \alpha_1 = {\boldsymbol{q}_{c_1}^3}^T \hat{\boldsymbol{x}}_w. \tag{46}$$

Similarly, for $\hat{\boldsymbol{x}}_{s_2}$ this is given as

$$\alpha_2 \hat{x}_{s_2} = {\boldsymbol{q}_{c_2}^1}^T \hat{\boldsymbol{x}}_w,\ \alpha_2 \hat{y}_{s_2} = {\boldsymbol{q}_{c_2}^2}^T \hat{\boldsymbol{x}}_w \text{ and } \alpha_2 = {\boldsymbol{q}_{c_2}^3}^T \hat{\boldsymbol{x}}_w. \tag{47}$$

Using the third relation of Equations (46) and (47) to remove the scaling variables, α_1 and α_2, the following relation is obtained.

$$\begin{bmatrix} \hat{x}_{s_1} {\boldsymbol{q}_{c_1}^3}^T - {\boldsymbol{q}_{c_1}^1}^T \\ \hat{y}_{s_1} {\boldsymbol{q}_{c_1}^3}^T - {\boldsymbol{q}_{c_1}^2}^T \\ \hat{x}_{s_2} {\boldsymbol{q}_{c_2}^3}^T - {\boldsymbol{q}_{c_2}^1}^T \\ \hat{y}_{s_2} {\boldsymbol{q}_{c_2}^3}^T - {\boldsymbol{q}_{c_2}^2}^T \end{bmatrix} \hat{\boldsymbol{x}}_w = \boldsymbol{S} \hat{\boldsymbol{x}}_w = 0 \tag{48}$$

$\hat{\boldsymbol{x}}_w$ is computed, using singular value decomposition, as the unit eigenvector of $\boldsymbol{S}^T \boldsymbol{S}$ associated with the smallest eigenvalue [65].

2.9. Displacement Transformation

Consider a single subset pair and its two sets of in-plane displacements determined by temporal matching between the reference and deformed image pairs under consideration. First $\boldsymbol{x}_{s_j}^d = \begin{bmatrix} x_{s_j}^d & y_{s_j}^d \end{bmatrix}^T$ is calculated based on $\boldsymbol{x}_{s_j}^o = \begin{bmatrix} x_{s_j}^o & y_{s_j}^o \end{bmatrix}^T$ as

$$\begin{bmatrix} x_{s_j}^d \\ y_{s_j}^d \end{bmatrix} = \begin{bmatrix} x_{s_j}^o \\ y_{s_j}^o \end{bmatrix} + \begin{bmatrix} u_j \\ v_j \end{bmatrix}. \tag{49}$$

Thereafter $\boldsymbol{x}_{s_j}^o$ and $\boldsymbol{x}_{s_j}^d$ are undistorted to obtain $\breve{\boldsymbol{x}}_{s_j}^o = \begin{bmatrix} \breve{x}_{s_j}^o & \breve{y}_{s_j}^o \end{bmatrix}^T$ and $\breve{\boldsymbol{x}}_{s_j}^d = \begin{bmatrix} \breve{x}_{s_j}^d & \breve{y}_{s_j}^d \end{bmatrix}^T$ respectively. This is performed using non-linear, least-squares optimization since an exact solution for the inverse of Equation (3) does not exist because it requires determining the roots of a polynomial of degree greater than four [74]. Triangulation, represented by function Ψ, determines the 3D coordinates in the world CS to which the reference subset pair, $\hat{\boldsymbol{x}}_w^o = [\hat{x}_w^o \ \ \hat{y}_w^o \ \ \hat{z}_w^o]^T$, and investigated subset pair, $\hat{\boldsymbol{x}}_w^d = [\hat{x}_w^d \ \ \hat{y}_w^d \ \ \hat{z}_w^d]^T$, correspond as

$$\hat{\boldsymbol{x}}_w^o = \Psi(\breve{\boldsymbol{x}}_{s_1}^o, \breve{\boldsymbol{x}}_{s_2}^o) \text{ and } \hat{\boldsymbol{x}}_w^d = \Psi(\breve{\boldsymbol{x}}_{s_1}^d, \breve{\boldsymbol{x}}_{s_2}^d). \tag{50}$$

Finally, the displacements $\hat{u}_w$, $\hat{v}_w$, and $\hat{w}_w$ are determined as

$$\begin{bmatrix}\hat{u}_w\\ \hat{v}_w\\ \hat{w}_w\end{bmatrix}=\begin{bmatrix}\hat{x}_w^d\\ \hat{y}_w^d\\ \hat{z}_w^d\end{bmatrix}-\begin{bmatrix}\hat{x}_w^o\\ \hat{y}_w^o\\ \hat{z}_w^o\end{bmatrix}. \quad (51)$$

3. Implementation

ADIC3D's framework, presented in Appendix A, is invoked from MATLAB's command prompt as "`ProcData = ADIC3D(FileNames1, FileNames2, Mask, GaussFilt, StepSize,SubSize, SubShape, SFOrder, RefStrat, StopCritVal, WorldCTs, ImgCTs);`" with the input variables defined in Table 1.

Table 1. Required input variables of the ADIC3D framework.

Variable	Variable Description
`FileNames1`	Cell array of character vectors containing the image file names of the first image series. All images need to be the same size.
`FileNames2`	Cell array of character vectors containing the image file names of the second image series. 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 analyzed during correlation.
`GaussFilt`	Defines 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 $\{\texttt{FiltSize} \in \mathbb{N}\}$.
`StepSize`	Step size in pixels $\{\texttt{StepSize} \in \mathbb{N}\}$.
`SubSize`	Subset size in pixels $\{\texttt{SubSize} = 2k+1 \mid k \in \mathbb{N}\}$.
`SubShape`	Subset shape $\{\texttt{SubShape} \in \texttt{'Square'}, \texttt{'Circle'}\}$.
`SFOrder`	Dictates the SF order $\{\texttt{SFOrder} \in \mathbb{Z} \mid 0 \leq \texttt{SFOrder} \leq 2\}$.
`RefStrat`	Logical statement dictating reference image strategy (Section 2.6.1).
`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 distorted sensor CSs defined according to MATLAB's `estimateCameraParameters` function.

ADIC3D returns a structured array `ResultData` containing four main fields: structured variable `Stereo` holding stereo-matching data as detailed in Table 2; structured arrays `ProcData1` and `ProcData2`, detailed in Table 3, holding the processing data for temporal matching of subsets of the first and second image series respectively; and structured array `DispTrans` containing displacement transformation data detailed in Table 4.

Table 2. Accessing the stereo matching data contained in `ResultData.Stereo` for subset `q`.

Variable	Variable Description
`P(b,q)`	SFPs (`b` = 1 for u and `b` = 7 for v).
`C(q)`	ZNCC coefficient.
`Iter(q)`	Number of iterations until stopping criterion is satisfied (maximum of 100 iterations).
`StopVal(q)`	Final stopping criterion value for subset `q`.

Table 3. Accessing the processing data for temporal matching (`ResultData.ProcData1(d)` and `ResultData.ProcData2(d)`) for image `d` and subset number `q`.

Variable	Variable Description
`ImgName`	Deformed 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 of the relevant camera (`b` $=1$ for x^o and `b` $=2$ for y^o).
`P(b,q)`	SFPs (`b` $=1$ for u and `b` $=7$ for v).
`C(q)`	ZNCC coefficient.
`Iter(q)`	Number of iterations until stopping criterion is satisfied (maximum of 100 iterations).
`StopVal(q)`	Final stopping criterion value.

Table 4. Accessing the displacement transformation data for image `d` (contained in `Result-Data.DispTrans(d)`) for subset `q`.

Variable	Variable Description
`Xow(b,q)`	Reference subset position in the world CS (`b` $=1$ for $\hat{x}_w^o$, `b` $=2$ for $\hat{y}_w^o$, and `b` $=3$ for $\hat{z}_w^o$).
`Uw(b,q)`	Displacement in the world CS (`b` $=1$ for $\hat{u}_w$, `b` $=2$ for $\hat{v}_w$, and `b` $=3$ for $\hat{w}_w$).
`CamParams`	Calibration parameters (extrinsic, intrinsic, and radial distortion parameters).

Refer to Appendix B for brief discussion of utilizing parallel processing to reduce computation time of ADIC3D.

3.1. `ADIC3D` Function

The main function `ADIC3D`, outlined in Table 5, sets up the DIC problem and calls the appropriate subroutines. The output variables are preassigned in lines 9–12 to save input data and efficiently store computed data. `ADIC3D` allows subsets to be analyzed in a flexible manner by assigning `Xos`, `SubSize`, `SubShape`, and `SFOrder` on a per subset and per image basis. However, despite being capable of this, `ADIC3D` assigns the same `SubSize`, `SubShape`, and `SFOrder` to each subset as is conventional for DIC. Subsets containing invalid pixels, identified by `Mask`, are eliminated.

The subroutine `StereoMatch` is called to perform stereo matching. `StereoMatch` has input variables `n` (the total number of images in each image series), `ResultData`, `FileNames1`, `FileNames2`, and `StopCritVal`. It returns the subset positions in the FIS2 stored as `ProcData2(d).Xos` and stereo matching correlation results `P`, `C`, `Iter`, and `StopVal` stored in `Stereo`.

Temporal matching is performed on each image series separately in lines 15 and 17 using the `ImgCorr` subroutine. `ImgCorr`'s inputs are `n`, `ProcData1` or `ProcData2`, `FileNames1` or `FileNames2`, `RefStrat` and `StopCritVal`. It returns `P`, `C`, `Iter`, and `StopVal` for each subset throughout the image series stored in `ProcData1` or `ProcData2` for the first and second image series respectively.

Finally, subroutine `CSTrans` determines coordinates and displacements in the world CS based on inputs `n`, `ResultData`, `WorldCTs`, `ImgCTs`, and `RefStrat`. `CSTrans` returns `Xow`, `Uw`, and `CamParams` stored in `DispTrans`. Note that within the subroutines `ResultData` is shortened to `RD` and `ProcData` is shortened to `PD` where appropriate.

Table 5. `ADIC3D` algorithm summary.

Line Numbers	Task Performed
Lines 2–3	Compute image names of both image series;
Lines 4–5	Compute number of images and size of the first image in first image series;
Lines 6–7	Create regularly spaced reference subset positions, `Xos`;
Line 8	Remove subsets containing invalid pixels which are defined by `Mask`;
Line 9–12	Pre-assign `ResultData` structure array;
Line 13	Call subroutine `StereoMatch` to perform stereo matching;
Line 15	Call subroutine `ImgCorr` to perform temporal matching of first image series;
Line 17	Call subroutine `ImgCorr` to perform temporal matching of second image series;
Line 18	Call subroutine `CSTrans` to perform displacement transformation from the distorted sensor CSs to the world CS;

3.2. Correlation Implementation

Both stereo and temporal matching make use of correlation, detailed in Section 2.3, which is implemented using the three subroutines: `SubShapeExtract`, `SFExpressions`, and `SubCorr`. These subroutines are discussed briefly since they are identical to those used in ADIC2D [53].

`SubShapeExtract`, detailed in Table 6, determines f_i, $\nabla \boldsymbol{f}_i$ and $\Delta \boldsymbol{x}_i$ of a subset based on inputs `SubSize`, `SubShape`, `Xos`, F, ∇F, and `SubExtract`. Note that variables with subscript i refer to the full set of this variable for the subset (i.e., f_i refers to $f_i \forall i \in I$) throughout Section 3.2. ∇F is the light intensity gradient of the reference image whereas `SubExtract` is an anonymous function which extracts a square subset from a matrix based on the size and position of the subset. For `SubShape` defined as ′`Circle`′, `SubSize` specifies the diameter.

Table 6. `SubShapeExtract` algorithm summary.

Line Numbers	Task Performed
Line 2	**switch** `SubShape`;
Line 3	**case** `SubShape` = ′`Square`′, **do**
Line 4–6	Extract f_i and $\nabla \boldsymbol{f}_i$ using `SubExtract`;
Line 7	Compute $\Delta \boldsymbol{x}_i$ using `SubSize`;
Line 8	**case** `SubShape` = ′`Circle`′, **do**
Line 9–11	Extract f_i and $\nabla \boldsymbol{f}_i$ using `SubExtract`;
Line 12	Compute $\Delta \boldsymbol{x}_i$ 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**

`SFExpressions` returns anonymous functions for expressions that are dependent on the SF order, as outlined in Table 7. These anonymous functions are as follows: `W` defines Equation (9); `dFdWdP` defines $\nabla \boldsymbol{f}_i \frac{\partial \boldsymbol{W}_i}{\partial \boldsymbol{P}}$; `SFPVec2Mat` defines Equation (22); `Mat2SFPVec` extracts the SFPs from `SFPVec2Mat`; and `StopCrit` defines Equation (23).

ADIC3D allows for the zero, first, or second-order SFs to be used by setting `SFOrder` to 0, 1, or 2 respectively.

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**

`SubCorr` performs the core operations of correlation for a subset as outlined in Table 8. Its inputs are the interpolation coefficients of the deformed image, f_i, $\nabla \boldsymbol{f}_i$, `SubSize`, `SFOrder`, `Xos`, $\Delta \boldsymbol{x}_i$, initial estimates for `P` and `StopCritVal`. `SubCorr` returns `P`, `C`, `Iter`, and `StopVal`. The size of vector `P` is consistent with that of the second-order SF. Thus, the elements of $\boldsymbol{P}^{SF2}$, not relevant to the specified SF order, remain zero.

Table 8. `SubCorr` algorithm summary.

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

`SubShapeExtract` is called prior to calling `SubCorr` in order to determine data used by `SubCorr` while `SFExpressions` is called within `SubCorr` to define the appropriate anonymous functions on a subset basis. Thus, the effect of the subset shape and SF order as well as the core mathematical operations of correlation can be considered separately.

3.3. Stereo Matching Implementation

Excluding the subroutines used to perform correlation, stereo matching is implemented using two additional subroutines: `FeatureMatch` and `StereoMatch`. `FeatureMatch` returns SFP initial estimates, determined according to the feature matching method outlined in Section 2.6.2, based on inputs `ProcData1`, `d` (the image to be analyzed), `F`, `G`, and `SubExtract`. `StereoMatch` refines these SFPs through correlation and uses them to determine the subset positions in the FIS2.

3.3.1. `StereoMatch` Function

`StereoMatch`, detailed in Table 9, loads the FIS1 and FIS2 as `F` and `G`, respectively, and performs Gaussian filtering on these using MATLAB's `imgaussfilt` function. Lines 4 and 6 can be changed to use alternative filtering methods. Bi-cubic b-spline interpolation coefficients are determined for the deformed image using MATLAB's `griddedInterpolant` function. In line 7 '`spline`' can be replaced with '`linear`' or '`cubic`' for bi-linear or bi-cubic polynomial interpolation respectively. Moreover, line 7 can be replaced with an alternative interpolation method such as MATLAB's `spapi` function. Line 9 of `SubCorr` may need to be updated if alternative interpolation methods are employed.

Lines 11 to 16 cycle through each subset of the FIS1, for which `FeatureMatch` in line 9 determined SFP initial estimates, performing correlation using subroutines `SubShapeExtract` and `SubCorr`. These displacement SFPs are used to determine the reference subset positions in the second image series.

Subset pairs which achieve a ZNCC coefficient below 0.6, or which fall outside the bounds of the FIS2, are assumed to have failed stereo matching. These subsets are determined in lines 21–22 and their subset positions in both image series are set to NaN (not a number) in lines 23–26 such that they are not analyzed during temporal matching or displacement transformation.

`StereoMatch` returns `ResultData` with the subset positions in the FIS2 assigned to `ProcData2(d).Xos`; the correlation results for stereo matching (`P`, `C`, `Iter`, `StopVal`) are assigned to `Stereo`.

Table 9. `StereoMatch` algorithm summary.

Line Numbers	Task Performed
Line 2	Define `SubExtract` function to extract square subset data;
Line 3–6	Load the FIS1 and FIS2 and perform Gaussian filtering on them;
Line 7	Compute interpolation coefficients of `G` using MATLAB's `griddedInterpolant` function;
Line 9	Call subroutine `FeatureMatch` to determine SFP initial estimates;
Line 10	Initialise temporary storage variables used to save correlation information during the for loop;
Line 11	**for** subset number `q` = `1` to number of subsets, **do**
Line 12	**if** `FeatureMatch` determined SFP initial estimates for subset `q`, **do**
Line 13	Call subroutine `SubShapeExtract`;
Line 14	Call subroutine `SubCorr`;
Line 15	**end if**
Line 16	**end for**
Line 17	Save correlation information to `RD.Stereo` variable;
Line 18	**for** image number `d` = `1` to `d` = `n`, **do**
Line 19	Compute the subset positions in the FIS2 using Equation (32);
Line 20	**end for**
Line 21–22	Determine subsets which fail stereo matching and subsets which pass;
Line 23–26	Set the subset position of subsets which fail stereo matching to NaNs throughout both image series;
Line 27	Display information for stereo matching;

3.3.2. `FeatureMatch` Function

`FeatureMatch`, outlined in Table 10, implements SIFT feature matching using codes available from the VLFeat open-source library of computer vision algorithms. More specifically, VLFeat's `vl_sift` function is used to determine the keypoint locations and descriptors of `F` and `G` in lines 3–4. The keypoints of `F` which fall within the perimeter of at least one subset (square subsets equivalent in size to that specified for the subset under consideration are used) are identified in line 5. Only these keypoints of `F` are retained, in line 6, in order to reduce the computation time of VLFeat's `vl_ubcmatch` function in determining matching keypoint pairs between `F` and `G` in lines 7–8.

For each subset, the 20 closest keypoints to its center position are identified in line 9. The for loop of line 13 cycles through the subsets using MSAC to determine reliable affine transformation parameters, which are used by Equation (31) to compute the corresponding second-order SFPs.

MSAC is implemented using MATLAB's `ransac` function. Two anonymous functions are used in the call to `ransac`. `RansacModel`, defined in line 10, determines the affine transformation parameters from the keypoint locations by solving Equation (29). `RansacError`, defined in line 11, evaluates Equation (30) to determine each keypoint's squared error distance (for keypoints identified to be relevant to the subset). The fifth argument of `ransac` sets $\tau = 1$ pixel^2 and the final argument specifies the confidence that the maximum number of inliers are determined. The fifth and sixth arguments of `ransac`, as well as the number of nearest keypoints used for each subset (line 9), can be altered to suit the reader's needs. The try statement of line 14 catches instances where `ransac` fails to identify three keypoint inliers. As such, the SFP initial estimates of these subsets, stored in `P`, remain NaN as initialized on line 12.

Table 10. `FeatureMatch` algorithm summary.

Line Numbers	Task Performed
Line 2	**if** VLFeat library is not setup, then return an error;
Line 3–4	Compute keypoint locations and descriptors for `F` and `G` using `vl_sift`;
Line 5	Determine vector `KptsInVacinity` identifying the keypoints of `F` which fall within the perimeter of the square subsets, equivalent in size to that specified for the subset under consideration, of `F`;
Line 6	Eliminate keypoints (and their associated descriptors) of `F` which do not fall within the perimeter of any of the subsets;
Line 7–8	Determine matching keypoints using `vl_ubcmatch`;
Line 9	Determine the 20 nearest keypoints for each subset stored in matrix `relevantKpts`;
Line 10–11	Define anonymous functions `RansacModel` and `RansacError` for determining affine transformation parameters (of Equation (29)) and evaluating Equation (30) respectively;
Line 12	Initialise `P` as NaNs;
Line 13	**for** subset number $q = 1$ to number of subsets, **do**
Line 14	**try**
Line 15	Use MSAC to determine affine transformation parameters for subset `q` from its relevant keypoints;
Line 16	Convert affine transformation parameters to second-order SFPs using Equation (31);
Line 17	**end try**
Line 18	**end for**
Line 19	Display information for SIFT feature matching;

3.4. Temporal Matching Implementation

Excluding the subroutines used to perform correlation, temporal matching is implemented using `PCM` and `ImgCorr`. ADIC3D's implementation of temporal matching is discussed briefly since it is almost identical to that of ADIC2D [53] with the exception that `PCM` and `ImgCorr` are modified to avoid analyzing subsets which fail stereo matching.

3.4.1. `PCM` Function

`PCM` determines an initial estimate of the displacement SFPs of a subset based on inputs `F`, `G`, `SubSize`, the subsets position, and `SubExtract` as outlined in Table 11. Subset positions are passed as separate x and y positions as required by MATLAB's `arrayfun` function which is used to call `PCM`. The if statement of line 2 avoids analyzing subsets which contain NaN positions and returns NaN displacements in this case.

Table 11. `PCM` algorithm summary.

Line Numbers	Task Performed
Line 2	**if** subset positions do not contain NaNs, **do**
Line 3	Compute normalized cross-power spectrum in the frequency domain;
Line 4	Convert back to spatial domain;
Line 5	Find index of the maximum correlation coefficient;
Line 6	Compute index vector which relates indices of the correlation coefficient matrix to the displacements they correspond to;
Line 7–8	Obtain displacements using index of the maximum correlation coefficient;
Line 9	**else, do**
Line 10	Set output displacements to NaN;
Line 11	**end if**

3.4.2. `ImgCorr` Function

`ImgCorr`, outlined in Table 12, makes use of a nested for loop to perform temporal matching. The outer loop cycles through the images of an image series, starting at the second image, preparing data for correlation. This includes loading the appropriate reference and deformed images; filtering the images; determining the interpolation coefficients of G; determining the reference image gradient; shifting the reference subset positions by the displacement SFPs determined in the previous correlation run (these displacement SFPs are rounded down such that the reference subset need not be interpolated [75]); and using `PCM` to determine initial estimates of the SFPs for the current correlation run. In the case of the absolute reference image strategy, the reference subset positions are not shifted and `PCM` is only used to obtain an initial estimate of the displacement SFPs for the first correlation run. For subsequent iterations, the SFPs of the previous correlation run are used as initial estimates.

The inner for loop cycles through the subsets calling `SubShapeExtract` and `SubCorr` to perform correlation of the subsets. The if statement of line 19 ensures that only subsets which pass stereo matching are analyzed within the for loop.

Table 12. `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 of G using MATLAB's `griddedInterpolant` function;
Line 7	**if** first image of correlation run **or** `RefStrat=1`, **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 displacement SFPs (incremental strategy);
Line 13	Call subroutine `PCM` to compute initial estimates of displacement SFPs;

Line 14	**else**, **do**
Line 15	Set `P(d)` ← `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	**if** subset `q` passed stereo matching, **do**
Line 20	Call subroutine `SubShapeExtract`;
Line 21	Call subroutine `SubCorr`;
Line 22	**end if**
Line 23	**end for**
Line 24	Save correlation information to `PD` variable;
Line 25–26	Display results for image `d` correlation;
Line 27	**end for**

3.5. Displacement Transformation Implementation

Displacement transformation is implemented using subroutines `CSTrans` and `Triangulation`.

3.5.1. `CSTrans` Function

`CSTrans`, outlined in Table 13, uses MATLAB's `estimateCameraParameters` function to determine the calibration parameters according to Section 2.2 and the fundamental matrix according to Section 2.4. The canonical form of the projection matrices are used such that the world CS is aligned with the camera CS of the first camera.

Lines 6–18 cycle through image pairs of the image set performing displacement transformation according to Section 2.9. Line 9 identifies subsets which do not contain NaN values in either their position or SFPs (subsets which pass stereo matching) which are to be analyzed during displacement transformation. MATLAB's `undistortPoints` function is used to remove distortion from the subset positions while the subroutine `Triangulation` uses polynomial and linear triangulation to determine the 3D coordinate in the world CS corresponding to a subset pair. The if statement of line 10 avoids recomputing the reference subset positions for the absolute reference image strategy.

`CSTrans` is susceptible to particular issues which can cause a fatal error. Refer to Appendix C for a discussion of a function which improves the robustness of `CSTrans`.

Table 13. `CSTrans` algorithm summary.

Line Numbers	Task Performed
Line 2	Compute calibration parameters using MATLAB's `estimateCameraParameters` function;
Line 3–4	Compute canonical projection matrices of first and second cameras;
Line 5	Load the fundamental matrix;
Line 6	**for** image number `d` = 1 to `d` = `n`, **do**
Line 7–8	Compute $\boldsymbol{x}_{s_1}^{d}$ and $\boldsymbol{x}_{s_2}^{d}$, Equation (49);
Line 9	Determine indices of subsets which do not contain NaNs in either $\boldsymbol{x}_{s_1}^{d}$ or $\boldsymbol{x}_{s_2}^{d}$;
Line 10	**if** first image pair of image set **or** `RefStrat` = 1, **do**
Line 11	Compute $\hat{\boldsymbol{x}}_{w}^{o}$ (for subsets identified in line 9) using the `Triangulation` subroutine and MATLAB's `undistortPoints` function;
Line 12	**else**
Line 13	Assign $\hat{\boldsymbol{x}}_{w}^{o}$ of previous image pair to current image pair;
Line 14	**end if**

Line 15	Compute $\hat{\boldsymbol{x}}_w^d$ (for subsets identified in line 9), using the `Triangulation` subroutine and MATLAB's `undistortPoints` function, and subtract $\hat{\boldsymbol{x}}_w^o$ from it to determine $\hat{u}_w$, $\hat{v}_w$ and $\hat{w}_w$ using Equation (51);
Line 16	Save calibration parameters;
Line 18	**end for**

3.5.2. `Triangulation` Function

`Triangulation`, outlined in Table 14, cycles through the subset pairs determining their ideal coordinates in the ideal sensor CSs using the polynomial triangulation method according to Section 2.7, in lines 3–20 (based on the code of Lourakis [76]), before performing linear triangulation according to Section 2.8, in lines 21–22, to determine their 3D coordinate in the world CS.

Table 14. `Triangulation` algorithm summary.

Line Numbers	Task Performed
Line 2	**for** subset pair number $q = 1$ to number of subsets, **do**
Lines 3–4	Determine translation matrices of Equation (35);
Line 5	Apply translation matrices to fundamental matrix according to Equation (36);
Lines 6–8	Compute epipoles and normalize them;
Lines 9–10	Compute rotation matrices according to Equation (37);
Line 11	Apply rotation matrices to fundamental matrix according to Equation (38);
Lines 12–13	Compute polynomial coefficients of Equation (43);
Lines 14–15	Compute real roots of the polynomial;
Line 16	Evaluate polynomial of Equation (42) at these roots;
Line 17	Determine root corresponding to minimum of Equation (42);
Lines 18–20	Compute ideal subset positions according to Equations (44) and (45);
Lines 21–22	Determine 3D position of noiseless subset positions using linear triangulation method;
Line 23	**end for**

4. Validation

The temporal matching aspect of ADIC3D has been shown to be sufficiently robust to contrast changes and noise while offering a compromise between noise suppression and spatial resolution consistent with that of established DIC algorithms [53]. Thus, Samples 1 and 2 of the Stereo-DIC Challenge [55], exhibiting rigid body translations which avoid introducing errors during temporal matching due to complex displacement fields, are used to assess the remaining aspects of ADIC3D in a quantitative manner. Sample 5, exhibiting complex displacement fields, assesses ADIC3D in a qualitative manner.

ADIC3D was used by setting `StopCritVal` $= 10^{-4}$ (limited to 100 iterations per subset), `FiltSize` $= 0$, `RefStrat` $= 0$, `SubShape` $=$ `'Square'`, `SFOrder` $= 1$, `SubSize` $= 61$, and `StepSize` $= 5$. A larger subset size was used to reduce errors during temporal matching. Increasing the subset size beyond 61 pixels produced only marginal improvements.

Additionally, DICe (Version 2.0-beta.16) [56] and LaVision's StrainMaster (version 1.3) were used to analyze the samples. DICe is well-established [77,78] and its modularity enables it to use the same subset positions in the FIS1 as ADIC3D prior to stereo matching. DICe was run using the same parameters as ADIC3D. StrainMaster, a streamlined version of LaVision's DaVis software, was used to analyze Samples 1 and 2. Its streamlined nature prevents modifying some correlation parameters such that it employed circular subsets, a stopping criterion of 0.01, and a maximum of 200 iterations. Although this prohibits its results from being directly comparable to ADIC3D, it is included to reflect the capabilities of a commercial code. All codes used identical masks for each sample.

For ADIC3D, subsets achieving a ZNCC criterion below 0.8 for stereo matching were eliminated while DICe and StrainMaster have their own methods of eliminating subsets which fail stereo matching. The final sets of valid subset pairs for ADIC3D and DICe were determined in two steps such that the results of the two codes are directly comparable.

First, a preliminary set of valid subset pairs was determined for each code as the subsets which achieved a ZNCC criterion above 0.8 for temporal matching. Additionally, for Samples 1 and 2 of DICe, Grubbs's test [79] was used to remove outliers, in terms of displacement error, to reduce outliers occurring in its preliminary set. Secondly, the final set of valid subset pairs for both codes was determined as the intersection of these preliminary sets based on the subset positions in the FIS1.

Successive images of all samples exhibited a jump in displacement such that `PCM` failed to determine SFP initial estimates. As such, subroutine `FeatureMatch` was used to obtain SFP initial estimates during temporal matching by inserting " `[PD(d).P]=FeatureMatch(PD,d,F,G, SubExtract);`" after line 16 in subroutine `ImgCorr`. DICe also used feature matching to determine SFP initial estimates.

The provided calibration image sets, of dot-grid-style calibration plates, were used by each code to perform calibration. ADIC3D used the `STEP1_CalcDLTparameters` function of MultiDIC, an open-source Stereo-DIC software proposed by Solav et al. [50], to locate the CTs in the distorted sensor CSs. All experimental image sets were captured using the FLIR Grasshopper 2 model Gras-50S5M cameras with (Edmund Optics) lenses having a focal length of 35 mm.

4.1. Samples 1 and 2

Sample 1 captures images of a complex specimen geometry, consisting of a flat base plate with protruding 3D features, as it is displaced in increments of 10 mm in the x- and z-directions according to Table A2 in Appendix D. The specimen is displaced by an Aerotech nano-positioning stage (model ANT130-160-XY) with a high-precision linear encoder and position feedback control such that the true displacements are known with high certainty. Sample 2 simulates the experiment of Sample 1 using the synthetic image generator documented by Balcaen et al. [62] to minimize error sources in the image set.

The world CS of each DIC code differs from that within which the displacements were imposed. Thus, for each code Pythagoras's theorem is used to compute the true, $\hat{\mu}_{w_q}^{true}$, and calculated, $\hat{\mu}_{w_q}^{calc}$, displacement magnitudes of each subset q. The resulting errors in displacement magnitude are quantified as bias, computed as the mean of the absolute error (MAE), and variance, computed as the standard deviation of the absolute error (σ).

$$MAE = \frac{\sum_{q=1}^{Q}\left|\hat{\mu}_{w_q}^{calc} - \hat{\mu}_{w_q}^{true}\right|}{Q} \qquad \sigma = \sqrt{\frac{\sum_{q=1}^{Q}\left(\left|\hat{\mu}_{w_q}^{calc} - \hat{\mu}_{w_q}^{true}\right| - MAE\right)^2}{Q-1}} \tag{52}$$

Here Q is the number of valid subset pairs in the image pair analyzed. For Samples 1 and 2, ADIC3D and DICe had 71,715 and 66,850 valid subsets, respectively, while StrainMaster had 112,021 and 108,085 valid subsets respectively. Tables 15 and 16 report errors metrics for Samples 1 and 2, respectively, for steps 3, 7, and 15 which correspond to displacement extremes.

Table 15. Displacement error metrics for Sample 1 reported at ×10^{-3} mm.

Step	ADIC3D		DICe		Strain Master	
	Bias	Variance	Bias	Variance	Bias	Variance
3	47.2	9.17	37.1	23	55.5	14.1
7	44.2	7.65	41.1	21.6	50.2	13
15	61.6	13.4	57.1	33.5	77.4	20.1

Table 16. Displacement error metrics for Sample 2 reported at ×10⁻³ mm.

Step	ADIC3D		DICe		Strain Master	
	Bias	Variance	Bias	Variance	Bias	Variance
3	1.55	2.48	5.14	4.59	11.3	5.89
7	1.4	2.16	4.48	4.4	3.07	6.47
15	2.35	3.52	9.92	6.86	7.03	21.2

The improved error metrics of Sample 2 relative to Sample 1 for each code, with the exception of StrainMaster's variance for step 15, reflects the impact of the reduced error sources of the synthetic image set. For Sample 1, ADIC3D performs on par with DICe (up to 27% higher bias) and StrainMaster (at least 12% lower bias). In contrast, for Sample 2 ADIC3D shows improved error metrics relative to DICe and StrainMaster (at least 69% and 54% lower bias respectively). StrainMaster's higher bias and variance are a result of it analyzing more subsets in the vicinity of protruding features, which are challenging to track, which are removed for ADIC3D and DICe.

4.2. Sample 5

Sample 5 captures images of an ASTM tensile specimen made of Aluminium 2024-T6 as it is loaded in tension. The displacements computed by ADIC3D and DICe are directly comparable since both align their world CS with the camera CS of the first camera.

Figure 4 shows the $\hat{u}_w$ displacement field of image pair 1000 superimposed on the FIS1. The mean and maximum of the percentage difference of the $\hat{u}_w$, $\hat{v}_w$, and $\hat{w}_w$ displacements, between ADIC3D and DICe, are reported in Table 17 for selected image pairs.

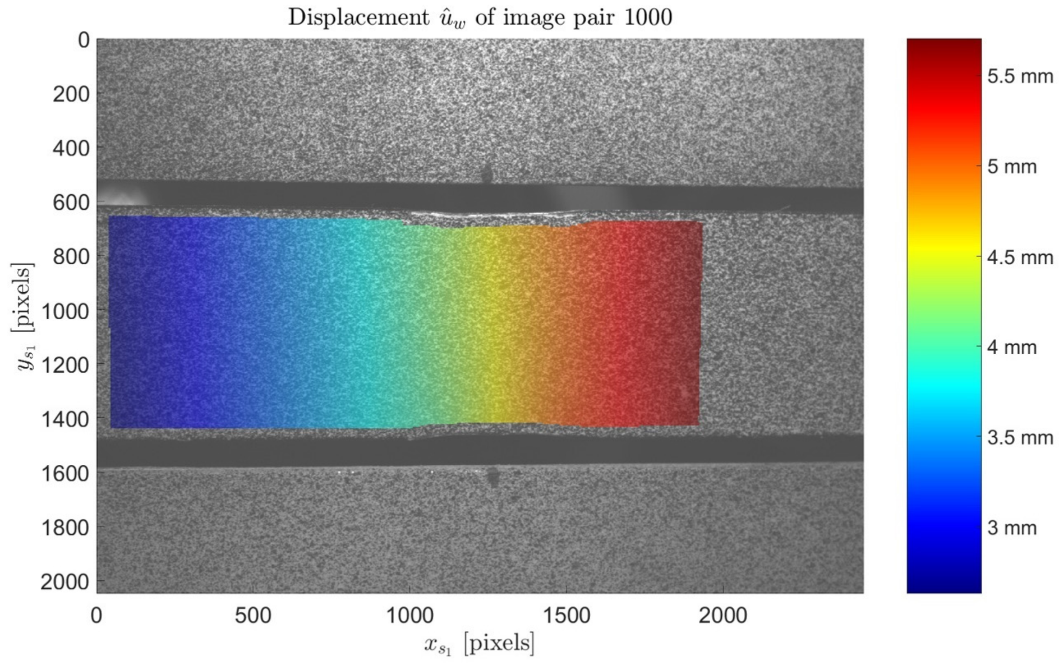

Figure 4. Displacement in the x-direction in the world CS superimposed on the FIS1 of the ASTM tensile specimen [55].

Table 17. Percentage difference of computed displacements between ADIC3D and DICe.

Image Pair	**250**		**500**		**750**		**1000**	
Metric	**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**
$\hat{u}_w$	0.0342	0.131	0.0354	0.113	0.0373	0.118	0.0374	0.122
$\hat{v}_w$	0.374	2.19	0.6	2.57	0.875	2.97	1.04	3.84
$\hat{w}_w$	0.093	0.443	0.193	0.744	0.291	0.926	0.345	1.02

The cameras being positioned apart horizontally in the x-direction is reasoned to cause the larger percentage difference of $\hat{v}_w$ and $\hat{w}_w$ relative to $\hat{u}_w$ which is consistent with the findings of Balcaen et al. [5]. The low percentage difference (remaining below 2% mean and 4% maximum) indicates that ADIC3D performs consistently with DICe.

5. Discussion

ADIC3D's modularity is threefold. Firstly, the main tasks are performed by separate subroutines, closely linked to the mathematical theory presented, allowing readers to progressively build up their understanding of ADIC3D. For instance, correlation is separated into three subroutines enabling the reader to consider: (i) how data is prepared based on the subset shape (`SubShapeExtract`); (ii) the SF order's influence on expressions for variables used during correlation (`SFExpressions`); and (iii) the core operations of correlation (`SubCorr`) separately. Furthermore, subroutines `ImgCorr` and `StereoMatch` show how correlation is utilized differently by temporal and stereo matching.

Secondly, the subset shape, SF order, interpolation method, and Gaussian filtering parameters can be readily changed. Although the influence of these on displacement results is well documented in experimental solid mechanics applications [3,4,45], this allows investigating their influence in novel applications to fine-tune ADIC3D's setup.

Thirdly, the SF order, subset size, and subset shape can be changed on a per subset and per image basis enabling straightforward integration with adaptive strategies. Adaptive strategies, which are gaining interest in the field of DIC, attempt to assign optimal parameters to each subset such that the resulting displacements are, theoretically, as reliable as possible and independent of the user.

In order for the code to retain its simplicity, four aspects, identified by Pan [41] as state-of-the-art requirements, were omitted. First, calibration does not use bundle adjustment, proposed by Furukawa and Ponce [80], to refine the calibration parameters. Additionally, Vo's method [81] of combing the frontal image plane and DIC to more accurately locate the CTs is not implemented. It should be noted that although the camera model presented in Section 2.2 only accounts for radial distortion, accounting for tangential distortion in ADIC3D is straightforward, as outlined in Appendix E.

Secondly, although the incremental reference image strategy is included, ADIC3D does not automatically employ it if the ZNCC coefficient falls below a certain threshold as recommended [82]. Thirdly, ADIC3D uses bi-cubic b-spline interpolation whereas bi-quintic b-spline interpolation is recommended [3] since increasing the degree of the interpolation method reduces the errors in the "ultimate error regime" [83] especially for smaller subsets.

Lastly, ADIC3D does not employ Pan's [84] reliability guided displacement tracking strategy (RGDT). ADIC3D's temporal matching uses the SFPs of each subset of the previous correlation run as initial estimates in the current correlation run. In contrast, RGDT only does this for the subset achieving the best ZNCC in the previous correlation run. This subset's optimized SFPs are used as an initial estimate to correlate its neighboring subsets. Of the correlated subsets, having uncorrelated neighbors, the one with the best ZNCC has its SFPs used as an initial estimate to correlate its neighbors. This is repeated to correlate all subsets in the current correlation run. As noted with ADIC2D [53], this exclusion makes ADIC3D susceptible to propagating spurious SFPs of a subset throughout the image series.

Furthermore, RGDT is excluded such that ADIC3D is path-independent (where each subset is correlated independently of its neighbors). Thus, this avoids the need to specify seed points and the issue path-dependent methods have, with complex specimen geometries, of transferring SFPs across subsets which experience different deformation modes. This also allows ADIC3D to leverage MATLAB's parfor loop to utilize parallel processing to reduce computation time (as discussed in Appendix B).

Additionally, although not a state-of-the-art requirement, ADIC3D's feature matching method can be improved. Sample 1 and 2 highlight its limitation as it determines invalid SFP initial estimates for subsets in the vicinity of protruding features that experience a large perspective change. Iniyan et al. [85] recommend using the DeepFlow [86] feature matching algorithm combined with the homography transform and MSAC.

Despite these omissions, ADIC3D's performance is on par with established codes. More specifically, Samples 1 and 2 show that ADIC3D determines displacement magnitudes in the world CS with bias and variance consistent with that of DICe and StrainMaster, while Sample 5 shows that ADIC3D determines displacement vectors within a 4% maximum difference of those determined by DICe. Although Samples 1 and 2 reveal limitations of ADIC3D's stereo matching, it performs sufficiently well in the presence of a complex specimen geometry.

This, coupled with ADIC2D's validation of temporal matching and thus the correlation method of ADIC3D indicates that ADIC3D performs sufficiently well for practical use in the field of experimental solid mechanics. Moreover, due to its modularity ADIC3D can be readily adapted to a wide range of applications across many fields. Furthermore, validation of this modular framework makes it both attractive as an education resource and as a basis to further the capabilities of DIC.

6. Conclusions

The theory of a subset-based, Stereo-DIC framework, that is predominantly consistent with current state-of-the-art techniques, and its implementation as a modular 202 line MATLAB code, was presented. This framework is an extension of the previously presented ADIC2D framework and as such is also comprised of square or circular subset shape selection, assigning of Gaussian filtering parameters, altering of the interpolation method, the zero, first and second-order SFs, reference image strategy selection, the Phase Correlation Method to determine SFP initial estimates for temporal matching and calibration through using MATLAB's image calibration toolbox. ADIC3D, being capable of performing Stereo-DIC to determine both in-plane and out-of-plane displacements in the world CS, is additionally comprised of stereo matching, SIFT feature matching to determine SFP initial estimates for stereo matching and displacement transformation using the polynomial and linear triangulation methods. Furthermore, ADIC3D allows for assignment of SF order, subset shape, and subset size on a per image and per subset basis enabling straightforward integration with adaptive strategies. The framework is modular, enabling the reader to gain a deeper understanding of DIC as well as encouraging readers to adapt ADIC3D for new applications. Validation of the full framework coupled with its modularity makes it appealing not only as an educational resource, but also as a starting point to develop the capabilities of DIC.

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. Both 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.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available in a publicly accessible repository that does not issue DOIs. Publicly available datasets were analyzed in this study. This data can be found here: [https://sem.org/3ddic].

Acknowledgments: The authors would like to thank P.L. Reu for information and advice received about using the Stereo-DIC Challenge.

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.

Glossary of Symbols

a_1 through a_6	Affine transformation parameters
α_j	Homogeneous scaling variable of the jth camera
$\boldsymbol{B}$	Fundamental matrix
$\boldsymbol{B}_1$	Fundamental matrix after translation of polynomial triangulation method
$\boldsymbol{B}_2$	Fundamental matrix after rotation of polynomial triangulation method
β	Gaussian filtering window size
c_{x_j} and c_{y_j}	Translation applied to sensor coordinate system of the jth camera
c_{s_j}	Skew of ideal sensor coordinate system of the jth camera
C_{ZNSSD}	Zero-mean normalized sum of squared difference correlation criterion
C_{ZNCC}	Zero-mean normalized cross correlation criterion
C_{ObjFun}	Objective function of correspondence problem
D	Geometric error cost function
δ	Function to determine the Euclidean distance between two points
$\underline{\boldsymbol{e}}_{s_j} = \begin{bmatrix} e_{s_j}^1 & e_{s_j}^2 & e_{s_j}^3 \end{bmatrix}^T$	Epipole of the jth camera
E_{proj}	Total projection error
E_k^{dist}	Squared error distance of kth keypoint
F	Reference image
$\boldsymbol{f}$	Reference subset
f_i	Light intensity value of pixel i of reference subset
$\bar{f}$	Mean light intensity value of reference subset
$\tilde{f}$	Normalization function of reference subset
$\nabla \boldsymbol{f}_i = \begin{bmatrix} \frac{\partial f_i}{\partial x} & \frac{\partial f_i}{\partial y} \end{bmatrix}$	Light intensity gradient of reference subset for pixel i
G	Deformed image
$\boldsymbol{g}$	Investigated subset
g_i	Light intensity value of pixel i of investigated subset
$\bar{g}$	Mean light intensity value of investigated subset
$\tilde{g}$	Normalization function of investigated subset
$\boldsymbol{H}$	Hessian of optimization equation
I	Number of pixels within a subset
$\boldsymbol{I}$	3×3 identity matrix
i	Subscript indicating that a variable is associated with the ith pixel of a subset
$\boldsymbol{J}$	Jacobian of optimization equation
j	Subscript indicating that a variable is associated with the jth camera
$\boldsymbol{K}_j$	Intrinsic parameter matrix of the jth camera
K	Number of keypoints under consideration
κ_j^1 and κ_j^2	Radial distortion parameters of the jth camera
L	Number of calibration targets within a calibration image series
$\underline{\boldsymbol{\lambda}}_j$	Epipolar line of the jth camera
$\boldsymbol{M}$	3×3 matrix
$\boldsymbol{m}$	3×1 vector
$[\boldsymbol{m}]_\times$	Skew symmetric matrix of $\boldsymbol{m}$
MAE	Mean of the absolute error

$\hat{\mu}_{w_q}^{true}$	True displacement magnitude
$\hat{\mu}_{w_q}^{calc}$	Calculated displacement magnitude
$\boldsymbol{\Omega}$	Homography applied to projection matrices
$\boldsymbol{\omega}$	Function to populate a square matrix with the shape function parameters
$\boldsymbol{P}$	Shape function parameters
$\Delta\boldsymbol{P}$	Iterative improvement of shape function parameters
$\boldsymbol{P}_{update}$	Updated shape function parameters
$\Vert\Delta\boldsymbol{P}\Vert$	Stopping criterion
Φ	Function representing conversion of coordinate from projective space to Euclidean space
Ψ	Function representing the triangulation methods
$\varphi_1 - \varphi_4$	Elements of the fundamental matrix for the polynomial triangulation method
$\boldsymbol{Q}_j$	Projection matrix of the jth camera
$\boldsymbol{Q}_{c_j}$	Projection matrix of the jth camera in canonical form
$\boldsymbol{q}_{c_j}^{n}$	The nth row of the canonical projection matrix of the jth camera
Q	Number of subset pairs per image pair
$\boldsymbol{R}_j$	Rotation matrix for pinhole camera model of the jth camera
$\boldsymbol{R}_j^{pm}$	Rotation matrix of the jth camera for the polynomial triangulation method
$\boldsymbol{S}$	Matrix for the linear triangulation method
σ^g	Standard deviation of Gaussian function
σ	Standard deviation of the absolute error
$\boldsymbol{T}_j$	Translation vector for pinhole camera model of the jth camera
$\boldsymbol{T}_j^{pm}$	Translation matrix of the jth camera for polynomial triangulation method
t	Variable for parameterization of epipolar lines
τ	Squared error distance threshold
u	Displacement in x-direction in the distorted sensor coordinate system
u_x, u_{xx}, u_y, u_{yy} and u_{xy}	Derivatives of the x-direction displacement
$\hat{u}_w$	Displacement in x-direction in the world coordinate system
$\boldsymbol{V}_j$	Extrinsic parameter matrix of the jth camera
v	Displacement in y-direction in the distorted sensor coordinate system
v_x, v_{xx}, v_y, v_{yy} and v_{xy}	Derivatives of the y-direction displacement
$\hat{v}_w$	Displacement in y-direction in the world coordinate system
$\boldsymbol{W}$	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
$\hat{w}_w$	Displacement in z-direction in the world coordinate system
$\hat{\boldsymbol{x}}_w = [\hat{x}_w \quad \hat{y}_w \quad \hat{z}_w]^T$	Ideal coordinate in the world coordinate system
$\hat{\boldsymbol{x}}_{s_j} = [\hat{x}_{s_j} \quad \hat{y}_{s_j}]^T$	Ideal coordinate in the ideal sensor coordinate system of the jth camera
$\hat{\boldsymbol{x}}_{n_j} = [\hat{x}_{n_j} \quad \hat{y}_{n_j}]^T$	Normalized ideal image coordinates of the jth camera
$\boldsymbol{x}_{n_j} = [x_{n_j} \quad y_{n_j}]^T$	Normalized distorted image coordinates of the jth camera
$\boldsymbol{x}_{s_j} = [x_{s_j} \quad y_{s_j}]^T$	Coordinate in the distorted sensor coordinate system of the jth camera
$\boldsymbol{x}_l^{true} = [x_l^{true} \quad y_l^{true}]^T$	True location of lth calibration target in distorted sensor coordinate system
$\boldsymbol{x}_l^{calc} = [x_l^{calc} \quad y_l^{calc}]^T$	Location of lth calibration target in distorted sensor coordinate system predicted by the camera model
$\boldsymbol{x}_i = [x_i \quad y_i]^T$	Pixel position of the ith pixel of the reference subset in distorted sensor coordinate system
$\boldsymbol{x}^o = [x^o \quad y^o]^T$	Center of reference subset in distorted sensor coordinate system
$\Delta\boldsymbol{x}_i = [\Delta x_i \quad \Delta y_i]^T$	Distance from the reference subset center to ith pixel position of reference subset

$\Delta\boldsymbol{x}'_i = [\Delta x'_i \quad \Delta y'_i]^T$	Distance from the reference subset center to i^{th} pixel position of investigated subset
$\boldsymbol{x}'_i = [x'_i \quad y'_i]^T$	i^{th} pixel position of the investigated subset in the distorted sensor coordinate system
$\boldsymbol{x}^o_{s_j} = \left[x^o_{s_j} \quad y^o_{s_j}\right]^T$	Reference subset position in distorted sensor coordinate system of the j^{th} camera
$\boldsymbol{x}^d_{s_j} = \left[x^d_{s_j} \quad y^d_{s_j}\right]^T$	Investigated subset position in the distorted sensor coordinate system of the j^{th} camera
$\breve{\boldsymbol{x}}^o_{s_j} = \left[\breve{x}^o_{s_j} \quad \breve{y}^o_{s_j}\right]^T$	Measured position of the reference subset in the ideal sensor coordinate system of the j^{th} camera
$\breve{\boldsymbol{x}}^d_{s_j} = \left[\breve{x}^d_{s_j} \quad \breve{y}^d_{s_j}\right]^T$	Measured position of the investigated subset in the ideal sensor coordinate system of the j^{th} camera
$\breve{\boldsymbol{x}}_{s_j} = \left[\breve{x}_{s_j} \quad \breve{y}_{s_j}\right]^T$	Measured coordinate in the ideal sensor coordinate system of the j^{th} camera
$\hat{\boldsymbol{x}}^o_w = [\hat{x}^o_w \quad \hat{y}^o_w \quad \hat{z}^o_w]^T$	Position of reference subset pair in the world coordinate system
$\hat{\boldsymbol{x}}^d_w = [\hat{x}^d_w \quad \hat{y}^d_w \quad \hat{z}^d_w]^T$	Position of investigated subset pair in the world coordinate system
$\boldsymbol{x}^k_{s_j} = \left[x^k_{s_j} \quad y^k_{s_j}\right]^T$	k^{th} keypoint location in the first image of the image series of the j^{th} camera
ξ_{x_j} and ξ_{y_j}	Scaling of metric units to pixels for the j^{th} camera
ζ	Maximum distance, along a single axis, from the center of the reference subset to a pixel in the reference subset

Appendix A. ADIC3D Framework Code

The code of the ADIC3D framework, presented below, can be accessed on GitHub at https://github.com/SUMatEng/ADIC3D (accessed on 1 May 2021).

```
function ResultData=ADIC3D(FileNames1,FileNames2,Mask,
GaussFilt,StepSize,SubSize,SubShape,SFOrder,RefStrat,
StopCritVal,WorldCTs, ImgCTs)
[~,ImNames1]=cellfun(@fileparts,FileNames1,'Uni',0);
[~,ImNames2]=cellfun(@fileparts,FileNames2,'Uni',0);
n=numel(FileNames1);
[r,c]=size(im2double(imread(FileNames1{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(:)']; clear 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);
ResultData.ProcData1=struct('ImgName', ImNames1,'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,1,size(Xos,2))},1,n),
'Xos',repmat({Xos},1,n),'P',repmat({zeros([12,size(Xos,2)])},
1,n),'C',repmat({NaN([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));
ResultData.ProcData2=struct('ImgName', ImNames2, '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,1,size(Xos,2))},1,n),
'Xos',repmat({Xos},1,n),'P',repmat({zeros([12,size(Xos,2)])},
1,n),'C',repmat({NaN([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));
ResultData.Stereo=struct('P',zeros([12,size(Xos,2)]), 'C',
NaN([1,size(Xos,2)]),'StopVal',ones([1,size(Xos,2)])*
StopCritVal, 'Iter',zeros([1,size(Xos,2)]));
```

```
ResultData.DispTrans=struct('Xow',repmat({NaN(3,size(Xos,2))},1,n),'Uw',repmat({NaN(3,size(Xos,2))},1,n),'CamParams',repmat({stereoParameters(cameraParameters,cameraParameters,zeros(3,3), zeros(1,3))},1,n))
ResultData=StereoMatch(n,ResultData,FileNames1,FileNames2,StopCritVal); % Section 2.6.2
fprintf('\nFirst image set...\n');
ResultData.ProcData1=ImgCorr(n,ResultData.ProcData1,FileNames1, RefStrat,StopCritVal); % Section 2.6.1
fprintf('\nSecond image set...\n');
ResultData.ProcData2=ImgCorr(n,ResultData.ProcData2,FileNames2, RefStrat,StopCritVal); % Section 2.6.1
ResultData=CSTrans(n,ResultData,WorldCTs,ImgCTs,RefStrat); % Section 2.9
```

```
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 [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 (9)
   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 (9)
   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 (9)
    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.^2,Zeta.^2,0.5*Zeta.^2,1,Zeta,Zeta,0.5*Zeta.^2,Zeta.^2
    ,0.5*Zeta.^2]).^2)); % Equation (23)
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
dfdWdP=dFdWdP(dX(:),dY(:),dfdx(:),dfdy(:));
Hinv=inv(dfdWdP'*dfdWdP); % inverse of Equation (19)
f_bar=mean(f(:)); f_tilde=sqrt(sum((f(:)-f_bar).^2)); %
Equations (13) and (14)
flag=0; Iter=1; dP=ones(size(P));
while flag==0
    [dXY]=W(dX(:),dY(:),P); % Equation (7)
    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)); %
    Equations (13) and (14)
    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 (12) substituted into Equation (15)
    else
        J=dfdWdP'*(f(:)-f_bar-f_tilde/g_tilde*(g(:)-g_bar)); %
        Summation of Equation (18)
        dP([1:SFOrder*3+0^SFOrder 7:6+SFOrder*3+0^SFOrder])=-
        Hinv*J; % Equation (18)
        P=Mat2SFPVec(SFPVec2Mat(P)/SFPVec2Mat(dP)); % Equation
        (21)
    end
    Iter=Iter+1;
end

function RD=StereoMatch(n,RD,ImNames1,ImNames2,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); tic;
F=im2double(imread(ImNames1{1}));
if all(RD.ProcData1(1).ImgFilt), F=imgaussfilt(F,
RD.ProcData1(1).ImgFilt(1),'FilterSize',
RD.ProcData1(1).ImgFilt(2)); end
G=im2double(imread(ImNames2{1}));
```

```
if all(RD.ProcData1(1).ImgFilt), G=imgaussfilt(G, RD.ProcData1(1).ImgFilt(1),‘FilterSize’, RD.ProcData1(1).ImgFilt(2)); end
InterpCoef=griddedInterpolant({1:1:size(G,1),1:1:size(G,2)}, G, ‘spline’);
[dFdx,dFdy]=imgradientxy(F, ‘prewitt’);
P=FeatureMatch(RD.ProcData1,1,F,G,SubExtract); % Section 3.3.2
C=NaN(1,size(P,2)); Iter=NaN(1,size(P,2)); StopVal=NaN(1,size(P,2));
for q=1:size(P,2) % can be changed to parfor for parallel processing
   if (sum(isnan(P(:,q)))==0)&&(sum(isnan(RD.ProcData1(1).Xos(:,q)))==0)
      [f,dfdx,dfdy,dX,dY]=SubShapeExtract(RD.ProcData1(1).SubSize(q),RD.ProcData1(1).SubShape(q,:),RD.ProcData1(1).Xos(:,q),F,dFdx,dFdy,SubExtract); % Section 3.2
      [Pout(:,q),C(q),Iter(q),StopVal(q)]=SubCorr(InterpCoef,f,dfdx,dfdy,RD.ProcData1(1).SubSize(q),RD.ProcData1(1).SFOrder(q),RD.ProcData1(1).Xos(:,q),dX,dY,P(:,q), StopCritVal); % Section 3.2
   end
end
RD.Stereo.P=Pout; RD.Stereo.C=C; RD.Stereo.Iter=Iter; RD.Stereo.StopVal=StopVal;
for d=1:n % determine subset positions in the FIS2 using Equation (32)
   RD.ProcData2(d).Xos(1,:)=RD.ProcData2(d).Xos(1,:)+round(RD.Stereo.P(1,:)); RD.ProcData2(d).Xos(2,:)=RD.ProcData2(d).Xos(2,:)+round(RD.Stereo.P(7,:));
end
FailedSubsetsCondition=(RD.Stereo.C>=0.6)==0|(RD.ProcData2(1).Xos(1,:)+(RD.ProcData2(1).SubSize(:)’-1)/2>size(G,2))|(RD.ProcData2(1).Xos(1,:)-(RD.ProcData2(1).SubSize(:)’-1)/2<1)|(RD.ProcData2(1).Xos(2,:)+(RD.ProcData2(1).SubSize(:)’-1)/2>size(G,1))|(RD.ProcData2(1).Xos(2,:)-(RD.ProcData2(1).SubSize(:)’-1)/2<1);
FailedSubsets=find(FailedSubsetsCondition); PassedSubsets=find(FailedSubsetsCondition==0);
for d=1:n
   RD.ProcData1(d).Xos(:,FailedSubsets)=NaN(2,size(FailedSubsets,2));
   RD.ProcData2(d).Xos(:,FailedSubsets)=NaN(2,size(FailedSubsets,2));
end
fprintf('Stereo results\t| Time (s)| CC (min) | CC (mean) | Iter (max)\n\t\t\t\t| %7.3f | % .5f | % .6f | %4.0f \nSubsets that failed stereo matching %d/%d\n',toc,min(RD.Stereo.C(PassedSubsets)),nanmean(RD.Stereo.C(PassedSubsets)),max(RD.Stereo.Iter(PassedSubsets)),size(FailedSubsets,2),size(P,2));
```

```
function [P]=FeatureMatch(PD,d,F,G,SubExtract)
if exist(‘vl_sift’)~=3, fprintf(‘\nError occurred, please setup the VLFeat library required for SIFT feature matching (algorithm can be found at: https://www.vlfeat.org\n’); end; time_before_sift=toc;
[xk1,d1] = vl_sift(im2single(uint8(255 * F)));
[xk2,d2] = vl_sift(im2single(uint8(255 * G)));
KptsInVacinity=((abs(PD(d).Xos(2,:)-xk1(2,:)’)<=PD(d).SubSize/2) +(abs(PD(d).Xos(1,:)-xk1(1,:)’)<=PD(d).SubSize/2))==2;
xk1=xk1(:,sum(KptsInVacinity,2)>=1); d1=d1(:,sum(KptsInVacinity,2)>=1);
```

```
[matches, scores] = vl_ubcmatch(d1, d2,1.25);
xk1=xk1(1:2,matches(1,:))’;   xk2=xk2(1:2,matches(2,:))’;
relevantKpts=knnsearch(xk1,PD(d).Xos’,’K’,20);
RansacModel=@(kpts) [[kpts(:,1) kpts(:,2) ones(size(kpts(:,1),1),1)]\kpts(:,3)-[1; 0; 0];[kpts(:,1) kpts(:,2) ones(size(kpts(:,1),1),1)]\kpts(:,4)-[0; 1; 0]]; % solves for affine transformation parameters of Equation (29)
RansacError=@(a, kpts) sum((kpts(:,3:4)’-[1+a(1), a(2), a(3); a(4), 1+a(5), a(6)]*[kpts(:,1)’; kpts(:,2)’;ones(1, size(kpts(:,1)’,2))]).^2,1); % Equation(30)
P=NaN(12,size(PD(d).Xos,2));
for q=1:size(PD(d).Xos,2) % can be changed to parfor for parallel processing
   try
   [a,~] = ransac([xk1(relevantKpts(q,:),:),xk2(relevantKpts(q,:),:)], @(data) RansacModel(data), @(model,data) RansacError(model,data), 3,1,’Confidence’, 99.5);
   P(:,q)=[a(1)*PD(d).Xos(1,q)+a(2)*PD(d).Xos(2,q)+a(3); a(1); a(2); 0; 0; 0;a(4)*PD(d).Xos(1,q)+a(5)*PD(d).Xos(2,q)+a(6); a(4); a(5); 0; 0; 0]; % Equation (31)
   end
end
fprintf(‘SIFT found %d matching keypoints in %5.2f seconds\n’,size(matches,2),toc-time_before_sift);
```

```
function [u,v]=PCM(F,G,SubSize,XosX,XosY,SubExtract)
if (isnan(XosX)==0)&&(isnan(XosY)==0)
   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=(ifft2(NCPS));
   [vid,uid]=find(CC==max(CC(:)));
   IndShift=-ifftshift(-fix(SubSize/2):ceil(SubSize/2)-1);
   u=IndShift(uid);
   v=IndShift(vid);
else
   u=NaN; v=NaN;
end
```

```
function PD=ImgCorr(n,PD,FileNames,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(FileNames{d}));
   if all(PD(d).ImgFilt), G=imgaussfilt(G,PD(d).ImgFilt(1),‘FilterSize’,PD(d).ImgFilt(2)); end
   InterpCoef=griddedInterpolant({1:1:size(G,1),1:1:size(G,2)},G, ‘spline’); % Section 2.3.2
   if any([RefStrat==1,d==2])
      F=im2double(imread(FileNames{d-1}));
      if all(PD(d).ImgFilt), F=imgaussfilt(F,PD(d).ImgFilt(1),’FilterSize’,PD(d).ImgFilt(2)); end
      [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 3.4.1
   else
      PD(d).P=PD(d-1).P;
   end
```

```
    P=NaN(size(PD(d).P)); C=NaN(size(PD(d).C)); Iter=NaN(size(PD(d).C)); StopVal=NaN(size(PD(d).C));
    for q=1:size(PD(d).Xos,2) % inner loop (can be changed to parfor for parallel processing)
       if (sum(isnan(PD(d).P(:,q)))==0)&&(sum(isnan(PD(d).Xos(:,q)))==0)
          [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
          [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
          end
     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 | %8.3f | %.6f | %.7f | %4.0f \n',d,n,toc,min(PD(d).C),nanmean(PD(d).C),max(PD(d).Iter));
end
```

```
function RD=CSTrans(n,RD,WorldCTs,ImgCTs,RefStrat)
CamParams=estimateCameraParameters(ImgCTs,WorldCTs,‘NumRadialDistortionCoefficients’,2); % Section 2.2
Q1=[CamParams.CameraParameters1.IntrinsicMatrix’,[0; 0; 0]]*[[eye(3), [0;0;0]]; 0, 0, 0, 1];
Q2=[CamParams.CameraParameters2.IntrinsicMatrix’,[0; 0; 0]]*[[CamParams.RotationOfCamera2’, CamParams.TranslationOfCamera2’]; 0, 0, 0, 1];
B=CamParams.FundamentalMatrix;
for d=1:n, tic
   Xds1=RD.ProcData1(d).Xos+[RD.ProcData1(d).P(1,:); RD.ProcData1(d).P(7,:)]; % Equation (49)
   Xds2=RD.ProcData2(d).Xos+[RD.ProcData2(d).P(1,:); RD.ProcData2(d).P(7,:)]; % Equation (49)
   indValid=find((isnan(Xds1(1,:))+isnan(Xds1(2,:))+isnan(Xds2(1,:))+isnan(Xds2(2,:)))==0);
   if d==1|RefStrat==1
      RD.DispTrans(d).Xow(:,indValid)=Triangulation(B,Q1,Q2,undistortPoints(RD.ProcData1(d).Xos(:,indValid)’,CamParams.CameraParameters1)’,undistortPoints(RD.ProcData2(d).Xos(:,indValid)’,CamParams.CameraParameters2)’);
   else
      RD.DispTrans(d).Xow=RD.DispTrans(d-1).Xow;
   end
   RD.DispTrans(d).Uw(:,indValid)=Triangulation(B,Q1,Q2,undistortPoints(Xds1(:,indValid)’,CamParams.CameraParameters1)’,undistortPoints(Xds2(:,indValid)’, CamParams.CameraParameters2)’)-RD.DispTrans(d).Xow (:,indValid); % Equation (51)
   RD.DispTrans(d).CamParams=CamParams;
   fprintf(‘CS transformation image: %d/%d\t\ttime:%.3f\n’,d,n,toc);
end
```

```
function [ptsOut]=Triangulation(B,Q1,Q2,pts1,pts2)
for i=1:size(pts1,2)
   T1inv=[1, 0, pts1(1,i); 0, 1, pts1(2,i); 0, 0, 1]; % Equation (35)
   T2inv=[1, 0, pts2(1,i); 0, 1, pts2(2,i); 0, 0, 1]; % Equation (35)
```

```
5.     B1=T2inv'*B*T1inv; % Equation (36)
6.     [U,~,V]=svd(B1,0);
7.     e1=V(:,3)./norm(V(1:2,3));
8.     e2=U(:,3)./norm(U(1:2,3));
9.     R1=[e1(1), e1(2), 0; -e1(2), e1(1), 0; 0, 0, 1]; % Equation
       (37)
10.    R2=[e2(1), e2(2), 0; -e2(2), e2(1), 0; 0, 0, 1]; % Equation
       (37)
11.    B2=R2*B1*R1'; % Equation (38)
12.    phi_1=B2(2,2); phi_2=B2(2,3); phi_3=B2(3,2); phi_4=B2(3,3);
13.    p=[- phi_4*phi_1^2*phi_3*e1(3)^4 +
       phi_2*phi_1*phi_3^2*e1(3)^4, phi_1^4 +
       2*phi_1^2*phi_3^2*e2(3)^2 - phi_1^2*phi_4^2*e1(3)^4 +
       phi_2^2*phi_3^2*e1(3)^4 + phi_3^4*e2(3)^4, 4*phi_1^3*phi_2
       - 2*phi_1^2*phi_3*phi_4*e1(3)^2 +
       4*phi_1^2*phi_3*phi_4*e2(3)^2 +
       2*phi_1*phi_2*phi_3^2*e1(3)^2 +
       4*phi_1*phi_2*phi_3^2*e2(3)^2 - phi_1*phi_2*phi_4^2*e1(3)^4
       + phi_2^2*phi_3*phi_4*e1(3)^4 + 4*phi_3^3*phi_4*e2(3)^4,
       6*phi_1^2*phi_2^2 - 2*phi_1^2*phi_4^2*e1(3)^2 +
       2*phi_1^2*phi_4^2*e2(3)^2 +
       8*phi_1*phi_2*phi_3*phi_4*e2(3)^2 +
       2*phi_2^2*phi_3^2*e1(3)^2 + 2*phi_2^2*phi_3^2*e2(3)^2 +
       6*phi_3^2*phi_4^2*e2(3)^4, - phi_1^2*phi_3*phi_4 +
       4*phi_1*phi_2^3 + phi_1*phi_2*phi_3^2 -
       2*phi_1*phi_2*phi_4^2*e1(3)^2 +
       4*phi_1*phi_2*phi_4^2*e2(3)^2 +
       2*phi_2^2*phi_3*phi_4*e1(3)^2 +
       4*phi_2^2*phi_3*phi_4*e2(3)^2 + 4*phi_3*phi_4^3*e2(3)^4, -
       phi_1^2*phi_4^2 + phi_2^4 + phi_2^2*phi_3^2 +
       2*phi_2^2*phi_4^2*e2(3)^2 + phi_4^4*e2(3)^4,
       phi_3*phi_2^2*phi_4 - phi_1*phi_2*phi_4^2];
14.    r=roots(p); % determine roots of polynomial of Equation
       (43)
15.    r=r(imag(r)==0);
16.    Ds=r.^2./(1+(r.*e1(3)).^2)+(phi_3.*r+phi_4).^2./((phi_1.*r
       +phi_2).^2+e2(3)^2*(phi_3.*r+phi_4).^2); % Equation (42)
17.    [t]=min(Ds);
18.    pts1temp=T1inv*R1'*[t^2*e1(3); t; t^2*e1(3)^2+1]; %
       Equation (44)
19.    pts2temp=T2inv*R2'*[e2(3)*(phi_3*t+phi_4)^2; -
       (phi_1*t+phi_2)*(phi_3*t+phi_4);
       (phi_1*t+phi_2)^2+e2(3)^2*(phi_3*t+phi_4)^2]; % Equation
       (45)
20.    ptsOut1=pts1temp(1:2)./pts1temp(3);
       ptsOut2=pts2temp(1:2)./pts2temp(3);
21.    [~,~,V]=svd([ptsOut1(1)*Q1(3,:)-Q1(1,:); ptsOut1(2)*
       Q1(3,:)-Q1(2,:); ptsOut2(1)*Q2(3,:)-Q2(1,:); ptsOut2(2)*
       Q2(3,:)-Q2(2,:)],0); % Section 2.8
22.    ptsOut(:,i)=V(1:3,4)./V(4,4);
23. end
```

Appendix B. Utilizing Parallel Processing for ADIC3D

Parallel processing can be used to reduce ADIC3D's execution time by making use of MATLAB's parfor loops. More specifically, the for loops in line 11 of `StereoMatch`, 13 of `FeatureMatch`, and 18 of `ImgCorr` can be replaced by parfor loops. It is for this reason that data within these for loops are saved to temporary storage variables, initiated prior to the for loops, and assigned to structured arrays thereafter. Note that parfor loops cause ADIC3D to require more random-access memory.

Appendix C. `UndistortPasser` Function

Requiring `CSTrans` to analyze a large number of subsets causes MATLAB's `undistortPoints` function to require a large amount of random-access memory. If it requires more memory than is available, this causes a fatal error. This is avoided by replacing calls to `undistortPoints` with calls to subroutine `UndistortPasser`, detailed in Table A1, which has inputs `Xos` and the calibration parameters and returns the undistorted subset positions. `UndistortPasser` passes batches of 100 subsets to `undistortPoints` in order to avoid this fatal error. Note that this also reduces the computation time of lines 11 and 15 of `CSTrans`.

Table A1. `UndistortPasser` algorithm summary.

Line Numbers	Task Performed
Line 2–3	Determine how many times the number of subsets is divisible by 100 and save as iterations;
Line 4	**for** `q` = 1 to iterations, **do**
Line 5	Undistort a batch of 100 subset positions using MATLAB's `undistortPoints` function;
Line 6	**end for**
Line 7–9	Undistort the remaining subsets, not processed in the for loop, using `undistortPoints`;

The `UndistortPasser` subroutine is give as

```
1.  function XosOut=UndistortPasser(Xos,CamParams)
2.  n=size(Xos,1);
3.  iterations=floor(n/100);
4.  for i=1:iterations
5.     XosOut((i-1)*100+1:i*100,:)=undistortPoints(Xos((i -
       1)*100+1:i*100,:),CamParams);
6.  end
7.  if rem(n,100)>0
8.     XosOut(iterations*100+1:iterations*100+rem(n,100),:)=
       undistortPoints(Xos(iterations*100+1:iterations*100+rem(n,
       100),:),CamParams);
9.  end
```

Appendix D. Imposed Displacements of Samples 1

Table A2. Mean and standard deviation of the imposed displacements measured by the stage's optical encoder for Sample 1 [55].

Step	$\hat{u}_w$ Mean (mm)	U STD (nm)	$\hat{w}_w$ Mean (mm)	W STD (nm)
1	0	7.01	0	6.76
2	0	7.69	10	6.16
3	0	6.30	20	6.21
4	0	7.67	−10	6.12
5	0	6.74	−20	6.33
6	−10	4.91	0	6.83
7	−20	5.71	0	7.27
8	10	6.53	0	6.79
9	20	5.69	0	7.37
10	−10	5.99	−10	4.57
11	−20	14.65	−20	25.19
12	10	7.65	10	6.43
13	20	6.10	20	6.54
14	−10	5.70	10	6.08
15	−20	5.14	20	6.45
16	10	6.29	−10	5.01
17	20	5.99	−20	6.07
18	0	6.36	0	7.59

Appendix E. Accounting for Tangential Distortion

Tangential distortion can be accounted for during the calibration and displacement transformation processes by altering line 2 of the `CSTrans` subroutine as "`CamParams= estimateCameraParameters(ImgCTs,WorldCTs, 'NumRadialDistortionCo-efficients',2, 'EstimateTangentialDistortion',true);`". MATLAB's `un-distortPoints` function automatically accounts for the tangential distortion model introduced.

References

1. 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, doi:10.1364/oe.16.007037.
2. Wang, Y.Q.; Sutton, M.A.; Bruck, H.A.; Schreier, H.W. Quantitative error assessment in pattern matching: Effects of intensity pattern noise, interpolation, strain and image contrast on motion measurements. *Strain* **2009**, *45*, 160–178, doi:10.1111/j.1475-1305.2008.00592.x.
3. 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, doi:10.1117/1.1314593.
4. Schreier, H.W.; Sutton, M.A. Systematic errors in digital image correlation due to undermatched subset shape functions. *Exp. Mech.* **2002**, *42*, 303–310, doi:10.1177/001448502321548391.
5. Balcaen, R.; Reu, P.L.; Lava, P.; Debruyne, D. Stereo-DIC uncertainty quantification based on simulated images. *Exp. Mech.* **2017**, *57*, 939–951, doi:10.1007/s11340-017-0288-9.
6. Wang, Y.Q.; Sutton, M.A.; Ke, X.D.; Schreier, H.W.; Reu, P.L.; Miller, T.J. On Error Assessment in Stereo-based Deformation Measurements. *Exp. Mech.* **2011**, *51*, 405–422, doi:10.1007/s11340-010-9449-9.
7. Hu, Z.; Xie, H.; Lu, J.; Wang, H.; Zhu, J. Error evaluation technique for three-dimensional digital image correlation. *Appl. Opt.* **2011**, *50*, 6239–6247, doi:10.1364/AO.50.006239.
8. Kammers, A.D.; Daly, S. Self-assembled nanoparticle surface patterning for improved digital image correlation in a scanning electron microscope. *Exp. Mech.* **2013**, *53*, 1333–1341, doi:10.1007/s11340-013-9734-5.
9. Jin, H.; Lu, W.Y.; Korellis, J. Micro-scale deformation measurement using the digital image correlation technique and scanning electron microscope imaging. *J. Strain Anal. Eng. Des.* **2008**, *43*, 719–728, doi:10.1243/03093247JSA412.
10. Rosu, A.M.; Pierrot-Deseilligny, M.; Delorme, A.; Binet, R.; Klinger, Y. Measurement of ground displacement from optical satellite image correlation using the free open-source software micmac. *ISPRS J. Photogramm. Remote Sens.* **2015**, *100*, 48–59, doi:10.1016/j.isprsjprs.2014.03.002.
11. Zhang, D.; Arola, D.D. Applications of digital image correlation to biological tissues. *J. Biomed. Opt.* **2004**, *9*, 691–700, doi:10.1117/1.1753270.
12. Palanca, M.; Tozzi, G.; Cristofolini, L. The use of digital image correlation in the biomechanical area: A review. *Int. Biomech.* **2016**, *3*, 1–21, doi:10.1080/23335432.2015.1117395.
13. Libertiaux, V.; Pascon, F.; Cescotto, S. Experimental verification of brain tissue incompressibility using digital image correlation. *J. Mech. Behav. Biomed. Mater.* **2011**, *4*, 1177–1185, doi:10.1016/j.jmbbm.2011.03.028.
14. 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, doi:10.1002/jbm.a.31268.
15. Genovese, K.; Lee, Y.U.; Lee, A.Y.; Humphrey, J.D. An improved panoramic digital image correlation method for vascular strain analysis and material characterization. *J. Mech. Behav. Biomed. Mater.* **2013**, *27*, 132–142, doi:10.1016/j.jmbbm.2012.11.015.
16. Masc, C.M.; Hoag, A.; Hoult, N.A.; Take, W.A. Field monitoring of a bridge using digital image correlation. *Proc. Inst. Civ. Eng. Bridg. Eng.* **2015**, 168, 3–12.
17. Yoneyama, S.; Kitagawa, A.; Iwata, S.; Tani, K.; Kikuta, H. Bridge deflection measurement using digital image correlation. *Exp. Tech.* **2007**, *31*, 34–40, doi:10.1111/j.1747-1567.2006.00132.x.
18. Winkler, J.; Specialist, C.; Hansen, M.D. Innovative long-term monitoring of the great belt bridge expansion joint using digital image correlation. *Struct. Eng. Int.* **2018**, *28*, 347–352, doi:10.1080/10168664.2018.1461539.
19. Yoneyama, S.; Ueda, H. Bridge deflection measurement using digital image correlation with camera movement correction. *Mater. Trans.* **2012**, *53*, 285–290.
20. Sousa, P.J.; Barros, F.; Lobo, P.; Tavares, P.J.; Moreira, P.M.G.P. Experimental measurement of bridge deflection using digital image correlation. *Procedia Struct. Integr.* **2019**, *17*, 806–811.
21. Ngeljaratan, L.; Moustafa, M.A. Structural health monitoring and seismic response assessment of bridge structures using target-tracking digital image correlation. *Eng. Struct.* **2020**, *213*, 110551, doi:10.1016/j.engstruct.2020.110551.
22. Tian, L.; Pan, B. Remote bridge deflection measurement using an advanced video deflectometer and actively illuminated LED targets. *Sensors* **2016**, *16*, 1344, doi:10.3390/s16091344.
23. Tung, S.H.; Shih, M.H.; Sung, W.P. Development of digital image correlation method to analyse crack variations of masonry wall. *Sadhana* **2008**, *33*, 767–779, doi:10.1007/s12046-008-0033-2.

24. Nghiem, H.L.; Al Heib, M.; Emeriault, F. Method based on digital image correlation for damage assessment in masonry structures. *Eng. Struct.* **2015**, *86*, 1–15, doi:10.1016/j.engstruct.2014.12.021.
25. Ghorbani, R.; Matta, F.; Sutton, M.A. Full-field deformation measurement and crack mapping on confined masonry walls using digital image correlation. *Exp. Mech.* **2015**, *55*, 227–243, doi:10.1007/s11340-014-9906-y.
26. Mazzanti, P.; Caporossi, P.; Muzi, R. Sliding time master digital image correlation analyses of cubesat images for landslide monitoring: The Rattlesnake Hills landslide (USA). *Remote Sens.* **2020**, *12*, 592, doi:10.3390/rs12040592.
27. Caporossi, P.; Mazzanti, P.; Bozzano, F. Digital image correlation (DIC) analysis of the 3 December 2013 Montescaglioso landslide (Basilicata, southern Italy): Results from a multi-dataset investigation. *ISPRS Int. J. Geo-Inf.* **2018**, *7*, 372, doi:10.3390/ijgi7090372.
28. 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, doi:10.3390/rs10060865.
29. Daehne, A.; Corsini, A. Kinematics of active earthflows revealed by digital image correlation and DEM subtraction techniques applied to multi-temporal LiDAR data. *Earth Surf. Process. Landf.* **2013**, *38*, 640–654, doi:10.1002/esp.3351.
30. Dridi, S.; Morestin, F.; Dogui, A. Use of digital image correlation to analyse the shearing deformation in woven fabric. *Exp. Tech.* **2012**, *36*, 46–52, doi:10.1111/j.1747-1567.2011.00776.x.
31. Hursa, A.; Rolich, T.; Ražić, S.E. Determining pseudo Poisson's ratio of woven fabric with a digital image correlation method. *Text. Res. J.* **2009**, *79*, 1588–1598, doi:10.1177/0040517509104316.
32. Helfrick, M.N.; Niezrecki, C.; Avitabile, P.; Schmidt, T. 3D digital image correlation methods for full-field vibration measurement. *Mech. Syst. Signal Process.* **2011**, *25*, 917–927, doi:10.1016/j.ymssp.2010.08.013.
33. Yu, L.; Pan, B. Single-camera high-speed stereo-digital image correlation for full-field vibration measurement. *Mech. Syst. Signal Process.* **2017**, *94*, 374–383, doi:10.1016/j.ymssp.2017.03.008.
34. Beberniss, T.J.; Ehrhardt, D.A. High-speed 3D digital image correlation vibration measurement: Recent advancements and noted limitations. *Mech. Syst. Signal Process.* **2017**, *86*, 35–48, doi:10.1016/j.ymssp.2016.04.014.
35. Rajaram, S.; Vanniamparambil, P.A.; Khan, F.; Bolhassani, M.; Koutras, A.; Bartoli, I.; Moon, F.; Hamid, A.; Benson Shing, P.; Tyson, J.; et al. Full-field deformation measurements during seismic loading of masonry buildings. *Struct. Control Health Monit.* **2017**, *24*, e1903, doi:10.1002/stc.1903.
36. 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, doi:10.2749/101686617X14881937384648.
37. 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, doi:10.1177/1475921713506766.
38. Peters, W.H.; Ranson, W.F. Digital imaging techniques in experimental stress analysis. *Opt. Eng.* **1982**, *21*, 427–431, doi:10.1117/12.7972925.
39. 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, doi:10.1007/BF02322488.
40. 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, doi:10.1007/BF02323555.
41. Pan, B. Digital image correlation for surface deformation measurement: Historical developments, recent advances and future goals. *Meas. Sci. Technol.* **2018**, *29*, 082001, doi:10.1088/1361-6501/aac55b.
42. 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, doi:10.1117/12.7973231.
43. Lu, H.; Cary, P.D. Deformation measurements by digital image correlation: Implementation of a second-order displacement gradient. *Exp. Mech.* **2000**, *40*, 393–400, doi:10.1007/BF02326485.
44. Tong, W. An evaluation of digital image correlation criteria for strain mapping applications. *Strain* **2005**, *41*, 167–175, doi:10.1111/j.1475-1305.2005.00227.x.
45. Pan, B. Bias error reduction of digital image correlation using gaussian pre-filtering. *Opt. Lasers Eng.* **2013**, *51*, 1161–1167, doi:10.1016/j.optlaseng.2013.04.009.
46. Baker, S.; Matthews, I. Lucas-kanade 20 years on: A unifying framework. *Int. J. Comput. Vis.* **2004**, *56*, 221–255, doi:10.1023/B:VISI.0000011205.11775.fd.
47. 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, doi:10.1007/s11340-016-0180-z.
48. Pan, B.; Li, K.; Tong, W. Fast, robust and accurate digital image correlation calculation without redundant computations. *Exp. Mech.* **2013**, *53*, 1277–1289, doi:10.1007/s11340-013-9717-6.
49. 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, doi:10.1016/j.optlaseng.2014.05.013.
50. Solav, D.; Moerman, K.M.; Jaeger, A.M.; Genovese, K.; Herr, H.M. MultiDIC: An open-source toolbox for multi-view 3D digital image correlation. *IEEE Access* **2018**, *6*, 30520–30535, doi:10.1109/ACCESS.2018.2843725.
51. Blaber, J.; Adair, B.; Antoniou, A. Ncorr: Open-source 2D digital image correlation matlab software. *Exp. Mech.* **2015**, *55*, 1105–1122, doi:10.1007/s11340-015-0009-1.

52. Jones, E.M.C.; Iadicola, M.A. A good practices guide for digital image correlation. *Int. Digit. Image Correl. Soc.* **2018**, *10*, doi:10.32720/idics/gpg.ed1.
53. Atkinson, D.; Becker, T. A 117 line 2D digital image correlation code written in matlab. *Remote Sens.* **2020**, *12*, 2906, doi:10.3390/rs12182906.
54. Sutton, M.A.; Yan, J.H.; Tiwari, V.; Schreier, H.W.; Orteu, J.J. The effect of out-of-plane motion on 2D and 3D digital image correlation measurements. *Opt. Lasers Eng.* **2008**, *46*, 746–757, doi:10.1016/j.optlaseng.2008.05.005.
55. Reu, P.; Wattrisse, B.; Wang, W.; Robert, L.; Bruck, H.; Daly, S.; Rodriguez-Vera, R.; Bugarin, F. Challenge Dataset: 3D-DIC Web Page. Available online: https://sem.org/content.asp?contentid=198 (accessed on 15 October 2020).
56. Turner, D.Z. Digital Image Correlation Engine (DICe) Reference Manual. Available online: http://dicengine.github.io/dice/index.html (accessed on 20 July 2020).
57. Bloomenthal, J.; Rokne, J. Homogeneous coordinates. *Vis. Comput.* **1994**, *11*, 15–26, doi:10.1007/BF01900696.
58. Bothe, T.; Li, W.; Schulte, M.; Von Kopylow, C.; Bergmann, R.B.; Jüptner, W.P.O. Vision ray calibration for the quantitative geometric description of general imaging and projection optics in metrology. *Appl. Opt.* **2010**, *49*, 5851–5860, doi:10.1364/AO.49.005851.
59. Dunne, A.K.; Mallon, J.; Whelan, P.F. Efficient generic calibration method for general cameras with single centre of projection. *Comput. Vis. Image Underst.* **2010**, *114*, 220–233, doi:10.1016/j.cviu.2009.05.005.
60. Wei, G.Q.; De Ma, S. Implicit and explicit camera calibration: Theory and experiments. *IEEE Trans. Pattern Anal. Mach. Intell.* **1994**, *16*, 469–480, doi:10.1109/34.291450.
61. Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 1330–1334, doi:10.1109/34.888718.
62. Balcaen, R.; Wittevrongel, L.; Reu, P.L.; Lava, P.; Debruyne, D. Stereo-DIC calibration and speckle image generator based on FE formulations. *Exp. Mech.* **2017**, *57*, 703–718, doi:10.1007/s11340-017-0259-1.
63. Hou, H.S.; Andrews, H.C. Cubic splines for image interpolation and digital filtering. *IEEE Trans. Acoust.* **1978**, *26*, 508–517, doi:10.1109/TASSP.1978.1163154.
64. Pan, B.; Xie, H.; Wang, Z. Equivalence of digital image correlation criteria for pattern matching. *Appl. Opt.* **2010**, *49*, 5501–5509, doi:10.1364/AO.49.005501.
65. Hartley, R.; Zisserman, A. *Multiple View Geometry in Computer Vision*, 2nd ed.; Cambridge University press: Cambridge, UK, 2004.
66. Hartley, R.I.; Sturm, P. Triangulation. *Comput. Vis. Image Underst.* **1997**, *68*, 146–157, doi:10.1006/cviu.1997.0547.
67. Foroosh, H.; Zerubia, J.B.; Berthod, M. Extension of phase correlation to subpixel registration. *IEEE Trans. Image Process.* **2002**, *11*, 188–200, doi:10.1109/83.988953.
68. Reu, P. Stereo-rig design: Stereo-angle selection—Part 4. *Exp. Tech.* **2013**, *37*, 1–2, doi:10.1111/ext.12006.
69. Garcia, D.; Orteu, J.J.; Penazzi, L. A combined temporal tracking and stereo-correlation technique for accurate measurement of 3D displacements: Application to sheet metal forming. *J. Mater. Process. Technol.* **2002**, *125*, 736–742, doi:10.1016/S0924-0136(02)00380-1.
70. Schreier, H.; Orteu, J.J.; Sutton, M.A. *Image Correlation for Shape, Motion and Deformation Measurements: Basic Concepts, Theory and Applications*; Springer: New York,NY, USA, 2009; ISBN 9780387787466.
71. Zhou, Y.; Pan, B.; Chen, Y.Q. Large deformation measurement using digital image correlation: A fully automated approach. *Appl. Opt.* **2012**, *51*, 6674–7683, doi:10.1364/ao.51.007674.
72. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110, doi:10.1023/B:VISI.0000029664.99615.94.
73. Torr, P.H.S.; Zisserman, A. MLESAC: A new robust estimator with application to estimating image geometry. *Comput. Vis. Image Underst.* **2000**, *78*, 138–156, doi:10.1006/cviu.1999.0832.
74. Żołądek, H. The topological proof of abel-ruffini theorem. *Topol. Methods Nonlinear Anal.* **2000**, *16*, 253–265, doi:10.12775/tmna.2000.040.
75. 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, doi:10.1016/j.optlaseng.2013.10.014.
76. Lourakis, M. Stereo Triangulation. Available online: https://www.mathworks.com/matlabcentral/fileexchange/67383-stereo-triangulation (accessed on 21 April 2020).
77. Turner, D.Z. An overview of the gradient-based local DIC formulation for motion estimation in DICe. *Sandia Rep.* **2016**, 1–6, doi:10.2172/1561808.
78. Lehoucq, R.B.; Reu, P.L.; Turner, D.Z. The effect of the ill-posed problem on quantitative error assessment in digital image correlation. *Exp. Mech.* **2017**, *61*, 609–621, doi:10.1007/s11340-017-0360-5.
79. Grubbs, F.E. Procedures for detecting outlying observations in samples. *Technometrics* **1969**, *11*, 1–21, doi:10.1080/00401706.1969.10490657.
80. Furukawa, Y.; Ponce, J. Accurate camera calibration from multi-view stereo and bundle adjustment. *Int. J. Comput. Vis.* **2009**, *84*, 257–268, doi:10.1007/s11263-009-0232-2.
81. Vo, M. Advanced geometric camera calibration for machine vision. *Opt. Eng.* **2011**, *50*, 110503, doi:10.1117/1.3647521.
82. Pan, B.; Dafang, W.; Yong, X. Incremental calculation for large deformation measurement using reliability-guided digital image correlation. *Opt. Lasers Eng.* **2012**, *50*, 586–592, doi:10.1016/j.optlaseng.2011.05.005.

83. 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, doi:10.1007/s11340-008-9204-7.
84. Pan, B. Reliability-guided digital image correlation for image deformation measurement. *Appl. Opt.* **2009**, *48*, 1535–1542, doi:10.1364/AO.48.001535.
85. Thiruselvam, N.I.; Subramanian, S.J. Feature-assisted stereo correlation. *Strain* **2019**, *55*, e12315, doi:10.1111/str.12315.
86. Weinzaepfel, P.; Revaud, J.; Harchaoui, Z.; Schmid, C. DeepFlow: Large displacement optical flow with deep matching. *Proc. IEEE Int. Conf. Comput. Vis.* **2013**, 1385–1392, doi:10.1109/ICCV.2013.175.