Computational Optimization of the 3D Least-Squares Matching Algorithm by Direct Calculation of Normal Equations

Frank Liebold * and Hans-Gerd Maas

Institute of Photogrammetry and Remote Sensing, Technische Universität Dresden, 01062 Dresden, Germany; hans-gerd.maas@tu-dresden.de
* Correspondence: frank.liebold@tu-dresden.de

Abstract: 3D least-squares matching is an algorithm that allows to measure subvoxel-precise displacements between two data sets of computed tomography voxel data. The determination of precise displacement vector fields is an important tool for deformation analyses in in-situ X-ray micro-tomography time series. The goal of the work presented in this publication is the development and validation of an optimized algorithm for 3D least-squares matching saving computation time and memory. 3D least-squares matching is a gradient-based method to determine geometric (and optionally also radiometric) transformation parameters between consecutive cuboids in voxel data. These parameters are obtained by an iterative Gauss-Markov process. Herein, the most crucial point concerning computation time is the calculation of the normal equations using matrix multiplications. In the paper at hand, a direct normal equation computation approach is proposed, minimizing the number of computation steps. A theoretical comparison shows, that the number of multiplications is reduced by 28% and the number of additions by 17%. In a practical test, the computation time of the 3D least-squares matching algorithm was proven to be reduced by 27%.

Keywords: 3D least-squares matching; cuboid tracking; 3D displacement field; voxel data; microtomography data

1. Introduction

Volume analysis in a voxel space representation is a logical extension of 2D image data processing. In the field of Photogrammetry and Computer Vision, image correlation techniques are often used for fine measurement of corresponding points in image sequences or stereo image pairs. Early contributions applied cross-correlation to image data [1]. In the early 80’s, gradient based algorithms were developed using an iterative least squares method [2] that also offers subpixel precision. Lucas and Kanade [2] only introduced two translation parameters, whereas Ackermann [3] and Grün [4] extended the model toward an affine and radiometric transformation, which also considers rotation, scaling and shear to obtain a better adaption for perspective distortions. As an example for the least-squares matching (LSM) algorithm, an image section of a textured concrete surface with two views is illustrated in Figure 1. Figure 1a shows a section of the reference image with the patch (subimage) consisting of the pixels included in the computation. The pixels are illustrated as red squares with red circles in the centers. In Figure 1b, an image section of the second view as well as the transformed patch (red squares with red circles) are depicted. In this example, the transformation includes translation, rotation, scale and shear and the corresponding parameters are obtained by the LSM algorithm. The most important parameters are the shifts, which typically show an accuracy of 0.02 to 0.05 pixels in this kind of application.
It is also possible to use more complex geometric models for the perspective transformation [5]. Grün and Baltsavias [6] introduced geometrically constrained multiphotograph matching, where the solution can inherently be forced to the epipolar line. Other contributions combined digital image matching and object surface reconstruction. In addition to a geometric 3D model of the surface, they used a radiometric model for the surface and the images. The unknowns of the surface densities, 3D points and image orientation parameters were computed in an iterative least-squares adjustment [7,8].

Maas et al. [9] transferred the idea from the 2D image matching to 3D voxel data (digital volume correlation DVC). They applied the subvoxel-precise cuboid tracking (3D least-squares matching) with 12 parameters of a 3D affine transformation to a sequence of multi-temporal voxel representations of mixing fluids. Bay et al. [10] extended the cross-correlation technique to 3D, combined it with a subvoxel-precise refinement and applied it to X-ray tomography data for strain analysis. Liebold et al. [11] also introduced a radiometric transformation to 3D cuboid matching.

In the last years, also free DVC code was provided. For instance, TomoWarp2 ensures the computation of 3D displacement fields using 3D cross-correlation [12]. They also implemented a further step to calculate subvoxel positions considering the maximum cross-correlation integer position and the 26 neighbor voxel around. The free software package SPAM is based on TomoWarp2 [13] and also provides a tool for the coarse registration that can consider a rotation between the reference and the deformed volume. In ALDVC [14], a Fast Fourier Transform (FFT) is applied for the estimation of initial shift values, followed by an algorithm including a 12-parameter affine transformation similar to 3D least-squares matching for computing subvoxel positions.

The least-squares matching algorithm contains an adjustment procedure where normal equations are computed. To speed up the process, Sutton et al. [15] showed a direct computation of the normal equations for 2D least-squares matching including affine parameters, and Jin et al. [16] also included radiometric parameters.

The publication at hand is based on the work of Maas et al. [9] and Liebold et al. [11]. It presents an optimized algorithm for the 3D LSM, likewise to what has been shown for 2D [15,16]. The central goal of the work is the acceleration of the 3D algorithm in order to save computation time and memory, without restrictions to the universality of the algorithm. The paper is structured as follows: In the next section, an overview of the computation of 3D least-squares matching is given including the optimization. In addition, a comparison to the standard method is shown. The publication closes with a conclusion.

2. 3D Least-Squares Matching

2.1. Mathematical Model

In the following, some formulae of Liebold et al. [11] are reused and extended. 3D least-squares matching is a gradient-based method and is applied to two voxel data sets. In
the reference volume, a small cuboid is defined containing the voxel of the cuboid’s center (point to be matched) and its neighborhood. The dimensions $dim_x, dim_y$ and $dim_z$ are odd numbers, and a typical cuboid size is for example $13 \times 13 \times 13 = 2197$ voxel. The aim is to find the corresponding cuboid in the second volume data set. The coordinates of the center of the cuboid in the reference state $x_{Ref,c}, y_{Ref,c}, z_{Ref,c}$ are integer values, whereas the coordinates of the voxel in the second state $x_{Def,c}, y_{Def,c}, z_{Def,c}$ are floating-point values.

Similar to Figure 1 for 2D, Figure 2 illustrates the 3D case with a voxel grid (gray). In the reference state, a subvolume consisting of e.g., $5 \times 5 \times 5$ voxel (green cuboid with red edges and red center points, Figure 2a). In the second state, depicted in Figure 2b, the affine-transformed cuboid as well as the faded reference subvolume are shown. The shift vector that connects the center points of the subvolumes is colored in magenta. 3D LSM is used to determine the transformation parameters, including the three translations.

![Figure 2](image)

**Figure 2.** (a) Reference voxel grid with the subvolume including the voxel (green with red edges and red center points) that are used for the computation. (b) Second volume with the transformed subvolume. The shift vector is colored in magenta and the reference subvolume is faded.

The range of the cuboid in the reference volume is:

$$
\begin{align*}
x_{Ref} & \in [x_{Ref,c} - h_x, x_{Ref,c} + h_x] \land x_{Ref} \in \mathbb{N} \\
y_{Ref} & \in [y_{Ref,c} - h_y, y_{Ref,c} + h_y] \land y_{Ref} \in \mathbb{N} \\
z_{Ref} & \in [z_{Ref,c} - h_z, z_{Ref,c} + h_z] \land z_{Ref} \in \mathbb{N}
\end{align*}
$$

where $h_x = \frac{dim_x - 1}{2}; h_y = \frac{dim_y - 1}{2}; h_z = \frac{dim_z - 1}{2}$.

Equation (2) shows the relationship between the gray values of the first (reference) and second (deformed) volume, also taking into account radiometric parameters ($r_0, r_1$). This relationship is valid for each voxel of the cuboid.

$$
f(x_{Ref}, y_{Ref}, z_{Ref}) = r_0 + r_1 \cdot g(x_{Def}, y_{Def}, z_{Def})
$$

where

- $x_{Ref}, y_{Ref}, z_{Ref} = \text{coordinates (reference state)}$
- $x_{Def}, y_{Def}, z_{Def} = \text{coordinates (deformed state)}$
- $f(x_{Ref}, y_{Ref}, z_{Ref}) = \text{gray value (reference state)}$
- $g(x_{Def}, y_{Def}, z_{Def}) = \text{gray value (deformed state)}$
- $r_0, r_1 = \text{brightness and contrast parameters}$

Between the reference and the deformed state, a coordinate transformation is performed. Similar to [3] in 2D and [9] in 3D, an affine transformation is used as mathematical model containing displacements, rotations, scaling and shear, see Equation (3), where $a_0, b_0$ and $c_0$ are the translation parameters.
\[
\begin{align*}
X_{\text{Def}} &= x_{\text{Ref},c} + a_0 + a_1 \cdot \bar{x} + a_2 \cdot \bar{y} + a_3 \cdot \bar{z} \\
Y_{\text{Def}} &= y_{\text{Ref},c} + b_0 + b_1 \cdot \bar{x} + b_2 \cdot \bar{y} + b_3 \cdot \bar{z} \\
Z_{\text{Def}} &= z_{\text{Ref},c} + c_0 + c_1 \cdot \bar{x} + c_2 \cdot \bar{y} + c_3 \cdot \bar{z}
\end{align*}
\] (3)

where
\[ a_i, b_i, c_i = \text{parameters of the affine transformation} \]
\[ \bar{x}, \bar{y}, \bar{z} = \text{reduced reference coordinates} \]

The reduced reference coordinates are computed as follows:
\[
\begin{align*}
\bar{x} &= x_{\text{Ref}} - x_{\text{Ref},c} \\
\bar{y} &= y_{\text{Ref}} - y_{\text{Ref},c} \\
\bar{z} &= z_{\text{Ref}} - z_{\text{Ref},c}
\end{align*}
\] (4)

Thus, the vector of unknowns is:
\[
p = (a_0, a_1, a_2, a_3, b_0, b_1, b_2, b_3, c_0, c_1, c_2, c_3, r_0, r_1)^T \] (5)

The parameter vector \( p \) consists of 14 unknowns, and a cuboid with an exemplary size of \( 13 \times 13 \times 13 = 2197 \) voxel (observations) leads to an over-determination. Thus, the parameters of \( p \) are computed in a least-squares adjustment process. Therefore, residuals are added to Equation (2) that results in the observation equations:
\[
f(x_{\text{Ref}}, y_{\text{Ref}}, z_{\text{Ref}}) + \nu(x_{\text{Ref}}, y_{\text{Ref}}, z_{\text{Ref}}) = r_0 + r_1 \cdot g(x_{\text{Def},0}, y_{\text{Def},0}, z_{\text{Def},0}) \]
\[
+ r_1 \cdot (g_{x,0} \cdot dx_{\text{Def}} + g_{y,0} \cdot dy_{\text{Def}} + g_{z,0} \cdot dz_{\text{Def}}) \] (6)

where \( v(x_{\text{Ref}}, y_{\text{Ref}}, z_{\text{Ref}}) = \text{residuum} \)

The first terms of the Taylor series expansion are used to linearize the observation equation, see Equation (7) where initial values of the parameters are included. They are marked with an additional zero subscript. Section 2.2 shows a way to obtain initial values for 3D LSM.

\[
f(x_{\text{Ref}}, y_{\text{Ref}}, z_{\text{Ref}}) + \nu(x_{\text{Ref}}, y_{\text{Ref}}, z_{\text{Ref}}) \approx r_{0,0} + d r_0 + (r_{1,0} + d r_1) \cdot g(x_{\text{Def},0}, y_{\text{Def},0}, z_{\text{Def},0}) \]
\[
+ r_1 \cdot (g_{x,0} \cdot dx_{\text{Def}} + g_{y,0} \cdot dy_{\text{Def}} + g_{z,0} \cdot dz_{\text{Def}}) \] (7)

where
\[ r_{0,0} = \text{initial value of the brightness parameter } r_0 \]
\[ r_{1,0} = \text{initial value of the contrast parameter } r_1 \]
\[ x_{\text{Def},0} = x \text{ position (deformed state) computed with the initial values } a_i \text{ of } a_i \]
\[ y_{\text{Def},0} = y \text{ position (deformed state) computed with the initial values } b_i \text{ of } b_i \]
\[ z_{\text{Def},0} = z \text{ position (deformed state) computed with the initial values } c_i \text{ of } c_i \]
\[ g_{x,0}, g_{y,0}, g_{z,0} = \text{volume gradients at the initial positions } x_{\text{Def},0}, y_{\text{Def},0}, z_{\text{Def},0} \]

The volume gradients of the gray values are written with their short forms in Equation (7). The full notation is:
\[
\begin{align*}
\partial g_{x,0} &= \frac{\partial g(x_{\text{Def}}, y_{\text{Def}}, z_{\text{Def}})}{\partial x_{\text{Def}}} \\
\partial g_{y,0} &= \frac{\partial g(x_{\text{Def}}, y_{\text{Def}}, z_{\text{Def}})}{\partial y_{\text{Def}}} \\
\partial g_{z,0} &= \frac{\partial g(x_{\text{Def}}, y_{\text{Def}}, z_{\text{Def}})}{\partial z_{\text{Def}}}
\end{align*}
\] (8)

There are different ways to obtain these gradients: One possibility is the computation of central differences \( s_{x,i}, s_{y,j}, s_{z,l} \) (numerical differentiation, Equation (9)) at integer positions \( x_{\text{Def},i}, y_{\text{Def},j}, z_{\text{Def},l} \) \((x_{\text{Def},i} \in \mathbb{N}, y_{\text{Def},j} \in \mathbb{N}, z_{\text{Def},l} \in \mathbb{N})\) combined with a tri-linear interpolation in order to calculate the derivatives at the floating point values of \( x_{\text{Def}}, y_{\text{Def}} \) and \( z_{\text{Def}} \).
\[ g_{x,i} \approx 0.5 \cdot \left( g(x_{\text{Def},i} + 1, y_{\text{Def},i}, z_{\text{Def},i}) - g(x_{\text{Def},i} - 1, y_{\text{Def},i}, z_{\text{Def},i}) \right) \]
\[ g_{y,i} \approx 0.5 \cdot \left( g(x_{\text{Def},i}, y_{\text{Def},i} + 1, z_{\text{Def},i}) - g(x_{\text{Def},i}, y_{\text{Def},i} - 1, z_{\text{Def},i}) \right) \]
\[ g_{z,i} \approx 0.5 \cdot \left( g(x_{\text{Def},i}, y_{\text{Def},i}, z_{\text{Def},i} + 1) - g(x_{\text{Def},i}, y_{\text{Def},i}, z_{\text{Def},i} - 1) \right) \]  

where

\[ g_{x,i} = \text{derivative in x direction at integer positions} \]
\[ g_{y,i} = \text{derivative in y direction at integer positions} \]
\[ g_{z,i} = \text{derivative in z direction at integer positions} \]

Another approach is the use of tri-cubic spline interpolation to compute the derivatives [17] (derivatives of cubic polynomials). And a further possibility is the calculation of the derivatives in the reference volume as an approximation. The advantage of the last point is once-only computation in the iterative least-squares process.

Thus, the differentials \( dx_{\text{Def}}, dy_{\text{Def}} \) and \( dz_{\text{Def}} \) are:

\[ dx_{\text{Def}} = \sum_{i=0}^{3} \frac{\partial x_{\text{Def}}}{\partial a_i} \cdot da_i = da_0 + \dot{x} \cdot da_1 + \ddot{y} \cdot da_2 + \dddot{z} \cdot da_3 \]
\[ dy_{\text{Def}} = \sum_{i=0}^{3} \frac{\partial y_{\text{Def}}}{\partial b_i} \cdot db_i = db_0 + \dot{x} \cdot db_1 + \ddot{y} \cdot db_2 + \dddot{z} \cdot db_3 \]
\[ dz_{\text{Def}} = \sum_{i=0}^{3} \frac{\partial z_{\text{Def}}}{\partial c_i} \cdot dc_i = dc_0 + \dot{x} \cdot dc_1 + \ddot{y} \cdot dc_2 + \dddot{z} \cdot dc_3 \]  

where

\[ dx_{\text{Def}} = \text{differential of the x coordinate} \]
\[ dy_{\text{Def}} = \text{differential of the y coordinate} \]
\[ dz_{\text{Def}} = \text{differential of the z coordinate} \]

The vector of corrections to the unknowns is:

\[ \mathbf{dp} = (da_0, \ da_1, \ da_2, \ da_3, \ db_0, \ db_1, \ db_2, \ db_3, \ dc_0, \ dc_1, \ dc_2, \ dc_3, \ dr_0, \ dr_1)^T \]  

The vector of unknowns results from the sum of the vector of the initial values \( \mathbf{p}_0 \) and the vector of the corrections to the unknowns \( \mathbf{dp} \):

\[ \mathbf{p} = \mathbf{p}_0 + \mathbf{dp} \]  

The linearized observation equations can be written in matrix notation:

\[ \mathbf{A} \cdot \mathbf{dp} = \mathbf{l} + \mathbf{v} \]  

where

\[ \mathbf{A} = \text{Jacobian matrix} \]
\[ \mathbf{l} = \text{reduced observation vector} \]
\[ \mathbf{v} = \text{residual vector} \]

The Jacobian matrix is built up as follows:

\[ \mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 & \mathbf{A}_3 & \mathbf{A}_4 \end{bmatrix} \]  

where
The individual residual $v_i$ is:

$$v_i = \sum_{i} (g_i - f_i)^2$$

The reduced observation vector $l$ is:

$$l = \begin{pmatrix}
    f_1 - r_0 - r_1 \cdot g_1 \\
    f_2 - r_0 - r_1 \cdot g_2 \\
    \vdots \\
    f_n - r_0 - r_1 \cdot g_n
\end{pmatrix}$$

In the Gauss-Markov model, the weighted sum of the squared residuals is minimized:

$$v^T \cdot W \cdot v \rightarrow \min_{dp}$$

where $W = \text{matrix of weights}$

In the following, only the case of equally weighted observations ($W = I$, $I$: identity matrix) is considered, see Equation (21).

$$v^T \cdot v \rightarrow \min_{dp}$$

The solution of this problem can be obtained using the normal equations:

$$A^T \cdot A \cdot dp = A^T \cdot l$$

As the computation of the normal matrix $A^T \cdot A$ and the right hand side $A^T \cdot l$ are done directly, it is not necessary to build up and store the whole Jacobian matrix $A$ and the reduced observation vector $l$. The detailed computation process is shown in Section 2.3.

After setting up the matrices $A^T \cdot A$ and $A^T \cdot l$, Equation (22) is solved for $dp$ and the initial values of the unknowns are updated:

$$p = p_0 + dp$$

The steps of the interpolation of the gray values and their derivatives as well as the computation of $A^T \cdot A$ and $A^T \cdot l$, the determination of $dp$ and the update of the unknowns are repeated until the process converges.

After the iterative adjustment routine, the residuals are determined by rearranging Equation (6). The gray values $g_i$ are obtained by applying Equation (3) with the updated affine parameters and tri-linear (or tri-cubic) interpolation. The individual residual $v_i$ is:
\[ v_i = r_0 + r_1 \cdot g_i (x_{Def}, y_{Def}, z_{Def}) - f_i (x_{Ref}, y_{Ref}, z_{Ref}) \]  

(24)

where \( v_i \) = individual residual

The standard deviation of the unit weight is:

\[ s_0 = \sqrt{\frac{\sum_{i=1}^{n} v_i^2}{n-u}} \]  

(25)

where

- \( s_0 \) = standard deviation of the unit weight
- \( n \) = number of voxel of the cuboid
- \( u \) = number of unknowns

2.2. Initial Values

The model in Equation (2) is not linear and if the movements between the volume data sets may exceed the dimensions of the cuboid, initial values have to be obtained for the 3D LSM algorithm.

Often, mainly translations and only small rotations occur between different epochs of measurements. In these cases, 3D cross-correlation can be used to obtain initial shifts [10]. An overview of the optimized algorithm can be found in Appendix A.

2.3. Direct Computation of the Normal Equations

As mentioned above, the normal matrix \((A^T \cdot A)\) and the right hand side vector \((A^T \cdot 1)\) are computed directly, so that building up the Jacobian matrix \(A\) as well as the reduced observation vector \(\mathbf{r}\) and the matrix multiplications can be avoided in order to gain efficiency and save memory. To compute the terms of the normal equations, the dot products of the columns of the Jacobian matrix (Equation (14)) as well as the dot products with the reduced observation vector are calculated (each combination of columns). Equation (26) shows the normal matrix that is organized in submatrices, \(G_{11,i}, G_{12,i}, G_{13,i}, G_{22,i}, G_{23,i}, G_{33,i}\) and \(G_{44,i}\) as well as the whole normal matrix are symmetric matrices and only the upper triangular matrices are shown. In Equation (28), the right hand side of the normal equations is depicted.

\[
A^T \cdot A = \begin{pmatrix}
A_1^T \cdot A_1 & A_1^T \cdot A_2 & A_1^T \cdot A_3 & A_1^T \cdot A_4 \\
A_2^T \cdot A_1 & A_2^T \cdot A_2 & A_2^T \cdot A_3 & A_2^T \cdot A_4 \\
A_3^T \cdot A_1 & A_3^T \cdot A_2 & A_3^T \cdot A_3 & A_3^T \cdot A_4 \\
A_4^T \cdot A_1 & A_4^T \cdot A_2 & A_4^T \cdot A_3 & A_4^T \cdot A_4 
\end{pmatrix} = \begin{pmatrix}
\sum_{i=1}^{n-1} G_{11,i} & \sum_{i=1}^{n-1} G_{12,i} & \sum_{i=1}^{n-1} G_{13,i} & \sum_{i=1}^{n-1} G_{14,i} \\
\sum_{i=1}^{n-1} G_{21,i} & \sum_{i=1}^{n-1} G_{22,i} & \sum_{i=1}^{n-1} G_{23,i} & \sum_{i=1}^{n-1} G_{24,i} \\
\sum_{i=1}^{n-1} G_{31,i} & \sum_{i=1}^{n-1} G_{32,i} & \sum_{i=1}^{n-1} G_{33,i} & \sum_{i=1}^{n-1} G_{34,i} \\
\sum_{i=1}^{n-1} G_{41,i} & \sum_{i=1}^{n-1} G_{42,i} & \sum_{i=1}^{n-1} G_{43,i} & \sum_{i=1}^{n-1} G_{44,i}
\end{pmatrix} 
\]  

(26)

where

- \( G_{11,i} = \begin{pmatrix} g_{11,ij} & g_{11,ik} & g_{11,il} \\
g_{11,ij} & g_{11,jk} & g_{11,jl} \\
g_{11,ij} & g_{11,ik} & g_{11,il} \end{pmatrix} \)
- \( G_{12,i} = \begin{pmatrix} g_{12,ij} & g_{12,ik} & g_{12,il} \\
g_{12,ij} & g_{12,jk} & g_{12,jl} \\
g_{12,ij} & g_{12,ik} & g_{12,il} \end{pmatrix} \)
- \( G_{13,i} = \begin{pmatrix} g_{13,ij} & g_{13,ik} & g_{13,il} \\
g_{13,ij} & g_{13,jk} & g_{13,jl} \\
g_{13,ij} & g_{13,ik} & g_{13,il} \end{pmatrix} \)
- \( G_{22,i} = \begin{pmatrix} g_{22,ij} & g_{22,ik} & g_{22,il} \\
g_{22,ij} & g_{22,jk} & g_{22,jl} \\
g_{22,ij} & g_{22,ik} & g_{22,il} \end{pmatrix} \)
- \( G_{23,i} = \begin{pmatrix} g_{23,ij} & g_{23,ik} & g_{23,il} \\
g_{23,ij} & g_{23,jk} & g_{23,jl} \\
g_{23,ij} & g_{23,ik} & g_{23,il} \end{pmatrix} \)
- \( G_{33,i} = \begin{pmatrix} g_{33,ij} & g_{33,ik} & g_{33,il} \\
g_{33,ij} & g_{33,jk} & g_{33,jl} \\
g_{33,ij} & g_{33,ik} & g_{33,il} \end{pmatrix} \)
- \( G_{44,i} = \begin{pmatrix} g_{44,ij} & g_{44,ik} & g_{44,il} \\
g_{44,ij} & g_{44,jk} & g_{44,jl} \\
g_{44,ij} & g_{44,ik} & g_{44,il} \end{pmatrix} \)

(27)
In fact, the geometric and radiometric parameters are usually independent on each other and are therefore often solved independently in LSM. If only the affine parameters are unknowns and the radiometric parameters are excluded from the model, the last two columns and last two rows of $A^T \cdot A$ as well as the last two values of $A^T \cdot 1$ are omitted. Appendix B and C consider the two cases concerning this point.

For each voxel, the following terms are computed and saved in temporal variables because they are used several times:

\[
\begin{align*}
\tilde{g}_{xx,i} &= g_{xx,i} \cdot \tilde{x}_i \\
\tilde{g}_{xy,i} &= g_{xy,i} \cdot \tilde{y}_i \\
\tilde{g}_{xz,i} &= g_{xz,i} \cdot \tilde{z}_i \\
\tilde{g}_{yx,i} &= g_{yx,i} \cdot \tilde{x}_i \\
\tilde{g}_{yy,i} &= g_{yy,i} \cdot \tilde{y}_i \\
\tilde{g}_{yz,i} &= g_{yz,i} \cdot \tilde{z}_i \\
\end{align*}
\]

(29)

Substituting the terms of Equation (29), the submatrices of the normal matrix and the right hand sight are shown in Equations (30) and (31). This reduces the number of computation steps, too.

\[
\text{\begin{align*}
\begin{pmatrix} G_{11,i} & G_{12,i} \\ G_{22,i} & G_{23,i} \\ G_{24,i} & G_{34,i} \\ G_{33,i} & G_{44,i} \end{pmatrix} &= \begin{pmatrix} g_{xx,i} \tilde{x}_i \cdot g_{xx,i} \tilde{x}_i \\
g_{xx,i} \tilde{x}_i \cdot g_{xx,i} \tilde{x}_i & g_{xx,i} \tilde{x}_i \cdot g_{xx,i} \tilde{x}_i \\
g_{xx,i} \tilde{x}_i \cdot g_{xx,i} \tilde{x}_i & g_{xx,i} \tilde{x}_i \cdot g_{xx,i} \tilde{x}_i \\
g_{xx,i} \tilde{x}_i \cdot g_{xx,i} \tilde{x}_i & g_{xx,i} \tilde{x}_i \cdot g_{xx,i} \tilde{x}_i \\
\end{pmatrix} + \begin{pmatrix} g_{xx,i} \tilde{x}_i \cdot g_{xx,i} \tilde{x}_i \\
g_{xx,i} \tilde{x}_i \cdot g_{xx,i} \tilde{x}_i & g_{xx,i} \tilde{x}_i \cdot g_{xx,i} \tilde{x}_i \\
g_{xx,i} \tilde{x}_i \cdot g_{xx,i} \tilde{x}_i & g_{xx,i} \tilde{x}_i \cdot g_{xx,i} \tilde{x}_i \\
g_{xx,i} \tilde{x}_i \cdot g_{xx,i} \tilde{x}_i & g_{xx,i} \tilde{x}_i \cdot g_{xx,i} \tilde{x}_i \\
\end{pmatrix}
\end{align*}
\]
\[ A^T \cdot 1 = \begin{pmatrix}
\sum_{n=1}^{m} f_n \cdot \sum_{i=1}^{n} y_{i,n} - \sum_{n=1}^{m} f_n \cdot \sum_{i=1}^{n} x_{i,n} - \sum_{n=1}^{m} f_n \cdot \sum_{i=1}^{n} r_i^2 - \sum_{n=1}^{m} f_n \cdot \sum_{i=1}^{n} S_{i,n}
\sum_{n=1}^{m} f_n \cdot \sum_{i=1}^{n} x_{i,n} - \sum_{n=1}^{m} f_n \cdot \sum_{i=1}^{n} y_{i,n} - \sum_{n=1}^{m} f_n \cdot \sum_{i=1}^{n} r_i^2 - \sum_{n=1}^{m} f_n \cdot \sum_{i=1}^{n} S_{i,n}
\sum_{n=1}^{m} f_n \cdot \sum_{i=1}^{n} y_{i,n} - \sum_{n=1}^{m} f_n \cdot \sum_{i=1}^{n} x_{i,n} - \sum_{n=1}^{m} f_n \cdot \sum_{i=1}^{n} r_i^2 - \sum_{n=1}^{m} f_n \cdot \sum_{i=1}^{n} S_{i,n}
\sum_{n=1}^{m} f_n \cdot \sum_{i=1}^{n} x_{i,n} - \sum_{n=1}^{m} f_n \cdot \sum_{i=1}^{n} y_{i,n} - \sum_{n=1}^{m} f_n \cdot \sum_{i=1}^{n} r_i^2 - \sum_{n=1}^{m} f_n \cdot \sum_{i=1}^{n} S_{i,n}
\end{pmatrix}\]  

(31)

2.4. Computational Effort

The computational costs are compared between the direct computation and the standard way that includes building up \( A \) and \( A^T \cdot 1 \). The detailed determination of the number of multiplication steps and addition steps is shown in the following.

2.4.1. Computational Costs of the Direct Method

A total of 100 terms have to be computed to form the normal equations directly, see Equation (32). This results from 60 terms of the matrices \( G_{11,i} \), \( G_{12,i} \), \( G_{13,i} \), \( G_{22,i} \), \( G_{23,i} \), \( G_{33,i} \) and 26 terms of the matrices \( G_{14,i} \), \( G_{24,i} \), \( G_{34,i} \) and \( G_{44,i} \) as well as 14 further terms of \( A^T \cdot 1 \). 26 terms (sums) of \( A^T \cdot 1 \) are already computed in \( G_{14,i} \), \( G_{24,i} \), \( G_{34,i} \) and \( G_{44,i} \).

To evaluate the computational effort, the number of additions and multiplications are counted:

- Multiplications per voxel:
  - Nine multiplications in terms of Equation (29).
  - 86 multiplications in the 100 terms (Equation (32), inside the sums).

- Additions per voxel:
  - 100 additions in the 100 sums (Equation (32)).

- Further multiplications:
  - Computation of the product \( r_1 \cdot r_1 \).
  - 84 multiplications of the prefactors with the sums in Equation (26) for the normal matrix.
  - 40 multiplications for the right hand side (Equation (31)) outside of the sums.

- Further additions:
28 additions for the right hand side (Equation (31)).

Altogether, for the direct computation of $A^T \cdot A$ and $A^T \cdot l$, 95 multiplications and 100 additions per voxel are required. The additional multiplication and summations for building up $A^T \cdot A$ and $A^T \cdot l$ can be neglected due to the large number of voxel.

2.4.2. Computational Costs of the Standard Method

The computational effort of the standard method contains building up the Jacobian matrix $A$ and the reduced observation vector $l$ as well as the matrix-matrix multiplication and the matrix-vector multiplication.

The following terms are calculated and saved in temporal variables for each line of the matrix $A$ (for each voxel):

$$
\begin{align*}
    r_{1gxi} &= r_1 \cdot g_{xi} \\
    r_{1gyi} &= r_1 \cdot g_{yi} \\
    r_{1gzi} &= r_1 \cdot g_{zi}
\end{align*}
$$

For the each line of matrix $A$, the following nine terms have to be computed:

$$
\begin{align*}
    r_{1gxi} \cdot \tilde{x}_i; r_{1gxi} \cdot \tilde{y}_i; r_{1gxi} \cdot \tilde{z}_i; r_{1gyi} \cdot \tilde{x}_i; r_{1gyi} \cdot \tilde{y}_i; r_{1gzi} \cdot \tilde{x}_i; r_{1gzi} \cdot \tilde{y}_i; r_{1gzi} \cdot \tilde{z}_i
\end{align*}
$$

For each element of $l$, there are one multiplication and two additions (subtractions):

$$
fi = f_0 - r_1 \cdot g_i
$$

• Multiplications per voxel:
  - One multiplication for building up the reduced observation vector $l$.
  - Three multiplications for the terms of Equation (33) that are computed temporally due to multiple use for building up the Jacobian matrix.
  - Nine further multiplications for building up the Jacobian matrix of Equation (34).
  - 105 multiplications (105 elements) due to the matrix multiplication of $A^T \cdot A$ (only upper triangular matrix).
  - 14 multiplications (14 elements) due to the matrix-vector multiplication of $A^T \cdot l$.

• Additions per voxel:
  - Two additions for building up the reduced observation vector $l$.
  - 105 additions (105 elements) due to the matrix multiplication of $A^T \cdot A$ (only upper triangular matrix).
  - 14 additions (14 elements) due to the matrix-vector multiplication of $A^T \cdot l$.

Altogether, for building up $A$ and $l$ and for the matrix multiplications $A^T \cdot A$ and $A^T \cdot l$, 132 multiplications and 121 additions per voxel are required.

2.4.3. Theoretical Comparison

Table 1 summarizes the consideration and shows the number of multiplications and additions per voxel for the standard method and the direct computation of the normal equations. The direct method reduces the number of multiplications by 28% and reduces the number of additions by 17%. Appendices B and C also consider the cases that the radiometric parameters are fixed and that these parameters are omitted. In addition to the reduced number of multiplications and summations, the direct method also save memory because the computation of the $n \times 14$ Jacobian matrix ($A$) and $n \times 1$ observation vector ($l$) is avoided.
Table 1. Comparison of the computational effort between the direct calculation of the normal equations and the standard method.

| Method   | Multiplications per vx | Additions per vx |
|----------|------------------------|-----------------|
| Standard | 132                    | 121             |
| Direct   | 95                     | 100             |

2.4.4. Comparison of Time Measurements in Practice

The comparison was done on a desktop machine with an AMD Ryzen 7 3800X 8-Cores (16 threads) processor at 3.9 GHz. In the test, 3D LSM was applied to 146,933 cuboids defined in the first two data sets of an in-situ tension test of Lorenzoni et al. [18]. The patch dimension was set to $15 \times 15 \times 15$ vx. The algorithm is implemented in C++ (using the GNU Compiler Collection, https://gcc.gnu.org) and compiled with the optimization option -O3. The loop over the matching points with the computation of the 3D LSM is parallelized with the OpenMP library (16 parallel units), and the time measurement only considers this loop. The time measurement includes the whole 3D LSM process so that not only the calculation of the normal matrix $A^T \cdot A$ and the right hand side vector $A^T \cdot l$ is analyzed. However, the computation of the normal equations is one of the most time-consuming steps herein.

Different calculation modes are tested: Method M1 stands for the direct computation. M2 represents the standard computation with building up the Jacobian matrix $A$ and the reduced observation vector $l$ as well as the matrix multiplications (only upper triangular matrix of the normal matrix) with an own implementation. M3 and M4 are similar to M2, but the matrix multiplications are done using the Eigen library. Eigen is a C++ template library for linear algebra (release 3.4.0, https://eigen.tuxfamily.org/). For M3, the full matrix of $A^T \cdot A$ is computed, and for M4, only the upper triangular matrix is computed. For all modes, before the calculation of the normal matrix begins, the values of the interpolated gray values and derivatives are computed and saved in separate arrays. All modes lead to the same numerical results, only the calculation times differ. Table 2 shows the calculation times for the different modes. The direct method is the fastest variant and almost twice as fast as the own standard computation M2. Compared to the best matrix multiplication variant M4 using Eigen (only upper triangular matrix), the direct method reduces the computation time by 27%. This reduction approximately agrees with the theoretical decrease of the multiplications of 28% (Section 2.4.3). The computation time difference between M3 and M4 is very small (only 9% decrease), although significantly more matrix elements (91) have to be computed (M3: 132 multiplications and 121 addition per voxel, M4: 223 multiplications and 212 addition per voxel).

Table 2. Comparison of the computation times with different modes for the calculation of the normal equations.

| Mode 1 | M1  | M2  | M3  | M4  |
|--------|-----|-----|-----|-----|
| Computation time [s] | 12.8 | 23.2 | 19.3 | 17.5 |

3. Conclusions

The paper presents an optimized algorithm to compute 3D least-squares matching in voxel data sequences including geometric and radiometric parameters. Compared to the standard method of the estimation of the normal equations, the number of multiplications is reduced by 28% and the number of additions is reduced by 17%. In a practical test, the computation time was decreased by 27%.
Author Contributions: Conceptualization, writing—review and editing, F.L. and H.-G.M.; methodology, writing—original draft preparation, visualization F.L.; supervision, funding acquisition, project administration, H.-G.M. All authors have read and agreed to the published version of the manuscript.

Funding: The research work presented in the publication has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation–SFB/TRR 280, Projekt-ID: 417002380).

Institutional Review Board Statement: Not applicable

Data Availability Statement: Not applicable

Acknowledgments: Furthermore, we thank the Institute of Construction Materials at Technische Universität Dresden, the Department of Chemical and Materials Engineering at the Pontificia Universidade Católica do Rio de Janeiro and the Bundesanstalt für Materialforschung und-prüfung in Berlin for providing the data sets.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

- LSM least-squares matching
- vx voxel
- DVC digital volume correlation
- SHCC strain-hardening cement-based composites
- FFT Fast Fourier Transform

Appendix A. Optimized 3D Cross-Correlation

In the following, an optimized method of the 3D cross-correlation is presented. In our approach, the normalized cross-correlation coefficient is used, see Equation (A2). \( t \) is the template, the subvolume of the reference volume, to be matched. The number of voxel of the template is

\[
n_{cc} = dim_x \cdot dim_y \cdot dim_z
\]  

(A1)

\[
\tilde{T}_{x+h_x,y+h_y,z+h_z} = \frac{\sum_{i,j,k} \tilde{g}_{x,y,z,i,j,k} \cdot \tilde{t}_{i,j,k}}{\sqrt{\sum_{i,j,k} \tilde{g}_{x,y,z,i,j,k}^2} \cdot \sqrt{\sum_{i,j,k} \tilde{t}_{i,j,k}^2}}
\]  

(A2)

where

\[
\tilde{g}_{x,y,z,i,j,k} = g_{x+i-1,y+j-1,z+k-1} - \tilde{g}_{x+h_x,y+h_y,z+h_z}
\]

\[
\tilde{t}_{i,j,k} = \bar{t}_{i,j,k} - \bar{t}
\]

\[
\bar{t} = \frac{1}{n_{cc}} \cdot \sum_{i,j,k} t_{i,j,k}; \text{mean intensity (template)}
\]

\[
\sum_{i,j,k} \rightarrow \sum_{i=1}^{dim_x} \sum_{j=1}^{dim_y} \sum_{k=1}^{dim_z}
\]

The terms of Equation (A2) are transformed as follows:

\[
\sum_{i,j,k} \tilde{g}_{x,y,z,i,j,k} \cdot \tilde{t}_{i,j,k} = \sum_{i,j,k} \left[ g_{x+i-1,y+j-1,z+k-1} \cdot t_{i,j,k} \right] - n_{cc} \cdot \bar{t} \cdot \tilde{g}_{x+h_x,y+h_y,z+h_z} + \bar{t} \cdot \tilde{g}_{x+h_x,y+h_y,z+h_z}
\]  

(A3)

\[
\sum_{i,j,k} \tilde{g}_{x,y,z,i,j,k}^2 = \sum_{i,j,k} g_{x+i-1,y+j-1,z+k-1}^2 - n_{cc} \cdot \bar{g}_{x+h_x,y+h_y,z+h_z}^2
\]  

(A4)

\[
\sum_{i,j,k} \tilde{t}_{i,j,k}^2 = \sum_{i,j,k} t_{i,j,k}^2 - 2 \cdot \sum_{i,j,k} t_{i,j,k} \cdot \bar{t} + \bar{t}^2 = \sum_{i,j,k} \tilde{t}_{i,j,k}^2 - n_{cc} \cdot \bar{t}^2
\]  

(A5)

There are optimized methods to compute the transformed terms using two approaches:
• The values of $\sum_{i,j,k} g_{x+i-1,y+j-1,z+k-1} \cdot t_{i,j,k}$ are computed for the whole search area using the 3D Fast Fourier transform as also done by Lewis [19] for 2D and Yang et al. [14] for 3D: $F^{-1}\{F(g) \circ F'(t)\}$ where $F'$ are the complex conjugate values ($\circ$: Hadamard product, element-wise multiplication). $t$ is a volume data set with the same size as the search volume containing $t$ and is filled up with zeros for indices exceeding the dimensions of $t$.

• The values of $\sum_{i,j,k} g_{x+i-1,y+j-1,z+k-1}^2$ and $\sum_{i,j,k} g_{x+i-1,y+j-1,z+k-1} t_{i,j,k}^2$ are computed using the integral volume method [20].

The terms $\sum_{i,j,k} t_{i,j,k}^2$ and $\bar{t}$ are computed straightforward. The position of the maximum coefficient $\max_{x,y,z} g_{x+h,y+h,z+h}$ can be used as initial position for the 3D LSM.

Appendix B. 3D Least-Squares Matching with Fixed Radiometric Parameters

This section considers the case that the radiometric parameters are fixed and shows the most important changes.

Appendix B.1. Mathematical Model

The relationship between the gray values of the reference and the second volume stays the same as before, see Equation (2).

$$f(x_{\text{Ref}}, y_{\text{Ref}}, z_{\text{Ref}}) = r_0 + r_1 \cdot g(x_{\text{Def}}, y_{\text{Def}}, z_{\text{Def}})$$ (A6)

The vector of unknowns reduces to the affine parameters:

$$\mathbf{p} = \begin{pmatrix} a_0, & a_1, & a_2, & a_3, & b_0, & b_1, & b_2, & b_3, & c_0, & c_1, & c_2, & c_3 \end{pmatrix}^T$$ (A7)

In the case of fixed radiometric parameters, the last two columns of the Jacobian matrix $\mathbf{A}$ are omitted compared to Equation (14) (for $\mathbf{A}_1, \mathbf{A}_2$ and $\mathbf{A}_3$ see Equations (15) to (17)):

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 & \mathbf{A}_3 \end{pmatrix}$$ (A8)

The reduced observation vector $\mathbf{l}$ does not change:

$$\mathbf{l} = \begin{pmatrix} f_1 - m - r_1 \cdot g_1 \\ f_2 - m - r_1 \cdot g_2 \\ \vdots \\ f_n - m - r_1 \cdot g_n \end{pmatrix}$$ (A9)

Thus, the normal matrix $\mathbf{A}^T \cdot \mathbf{A}$ and the right hand side $\mathbf{A}^T \cdot \mathbf{l}$ are also reduced compared to Equations (26) to (31):

$$\mathbf{A}^T \cdot \mathbf{A} = \begin{pmatrix} \mathbf{A}_1^T & \mathbf{A}_2^T & \mathbf{A}_3^T \end{pmatrix} \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 & \mathbf{A}_3 \end{pmatrix} = \begin{pmatrix} r_1^T \sum_{i=1}^n G_{11,i} & r_1^T \sum_{i=1}^n G_{12,i} & r_1^T \sum_{i=1}^n G_{13,i} \\ r_2^T \sum_{i=1}^n G_{21,i} & r_2^T \sum_{i=1}^n G_{22,i} & r_2^T \sum_{i=1}^n G_{23,i} \\ r_3^T \sum_{i=1}^n G_{31,i} & r_3^T \sum_{i=1}^n G_{32,i} & r_3^T \sum_{i=1}^n G_{33,i} \end{pmatrix}$$ (A10)

$$\mathbf{A}^T \cdot \mathbf{l} = \begin{pmatrix} \mathbf{A}_1 \cdot \mathbf{l} \\ \mathbf{A}_2 \cdot \mathbf{l} \\ \mathbf{A}_3 \cdot \mathbf{l} \end{pmatrix} = \begin{pmatrix} r_1^T \sum_{i=1}^n d f_{i,g_{x,i}} \\ r_2^T \sum_{i=1}^n d f_{i,g_{y,i}} \\ r_3^T \sum_{i=1}^n d f_{i,g_{z,i}} \\ r_1^T \sum_{i=1}^n d f_{i,gy_{x,i}} \\ r_2^T \sum_{i=1}^n d f_{i,gy_{y,i}} \\ r_3^T \sum_{i=1}^n d f_{i,gy_{z,i}} \\ r_1^T \sum_{i=1}^n d f_{i,gz_{x,i}} \\ r_2^T \sum_{i=1}^n d f_{i,gz_{y,i}} \\ r_3^T \sum_{i=1}^n d f_{i,gz_{z,i}} \end{pmatrix}$$ (A11)

where $df_i = f_i - r_0 - r_1 \cdot g_i$. 
Appendix B.2. Computational Effort

Appendix B.2.1. Direct Method

As above, for each voxel, the following terms are computed and saved in temporal variables because they are used several times:

\[
\begin{align*}
  g_{xx_i} &= g_{x,i} \cdot \bar{x}_i \\
  g_{xy_i} &= g_{x,i} \cdot \bar{y}_i \\
  g_{xz_i} &= g_{x,i} \cdot \bar{z}_i \\
  g_{yx_i} &= g_{y,i} \cdot \bar{x}_i \\
  g_{yy_i} &= g_{y,i} \cdot \bar{y}_i \\
  g_{yz_i} &= g_{y,i} \cdot \bar{z}_i \\
  g_{zx_i} &= g_{z,i} \cdot \bar{x}_i \\
  g_{zy_i} &= g_{z,i} \cdot \bar{y}_i \\
  g_{zz_i} &= g_{z,i} \cdot \bar{z}_i \\
  d_{fi} &= f_i - r_0 - r_1 \cdot g_i 
\end{align*}
\]  
(A12)

60 terms of the matrices \(G_{11}, G_{12}, G_{13}, G_{22}, G_{23}\) and \(G_{33}\) (Equation (30)) and 12 terms of \(A^T \cdot 1\) are computed. To evaluate the computational effort of the direct method, the number of additions and multiplications are counted. Only the operations per voxel are considered, the other operations can be neglected.

- **Multiplications per voxel:**
  - Ten multiplications for the temporally terms of Equation (A12).
  - 60 multiplications in the \(G_{ij}\) matrices.
  - 12 multiplications in the sums of \(A^T \cdot 1\).

- **Additions per voxel:**
  - Two additions (subtractions) in \(d_{fi} = f_i - r_0 - r_1 \cdot g_i\).
  - 60 additions in the 60 sums of the \(G_{ij}\) matrices.
  - 12 additions in the 12 sums of \(A^T \cdot 1\).

All in all, 82 multiplications and 74 additions have to be computed per voxel for the direct method.

Appendix B.2.2. Standard Method

For the standard method, building up \(A\) and \(I\) as well as the matrix multiplications are considered:

- **Multiplications per voxel:**
  - One multiplication for building up the reduced observation vector \(I\).
  - Three multiplications for the terms of Equation (33) that are computed temporally due to multiple use for building up the Jacobian matrix.
  - Nine further multiplications for building up the Jacobian matrix of Equation (34).
  - 78 multiplications (78 elements) due to the matrix multiplication of \(A^T \cdot A\) (only upper triangular matrix).
  - 12 multiplications (12 elements) due to the matrix-vector multiplication of \(A^T \cdot 1\).

- **Additions per voxel:**
  - Two additions (subtractions) for building up the reduced observation vector \(I\).
  - 78 additions (78 elements) due to the matrix multiplication of \(A^T \cdot A\) (only upper triangular matrix).
  - 12 additions (12 elements) due to the matrix-vector multiplication of \(A^T \cdot 1\).

Thus, for the standard method, 103 multiplications and 92 additions have to be computed per voxel.
Appendix B.2.3. Comparison

Table A1 shows the number of computations per voxel for the direct and the standard method. Using the direct method, 20.4% less multiplications and 19.6% less additions are done.

Table A1. Comparison of the computational effort between the direct computation of the normal equations and the standard method (affine mathematical model with fixed radiometric parameters).

| Method   | Multiplications per vx | Additions per vx |
|----------|-------------------------|------------------|
| Standard | 103                     | 92               |
| Direct   | 82                      | 74               |

Appendix C. 3D Least-Squares Matching with 12-Parameter Affine Transformation

In this section, only the affine transformation with 12 parameters is used as mathematical model.

Appendix C.1. Mathematical Model

The relationship between the gray values of the reference and the second volume is reduced to:

\[ f(x_{\text{Ref}}, y_{\text{Ref}}, z_{\text{Ref}}) = g(x_{\text{Def}}, y_{\text{Def}}, z_{\text{Def}}) \]  \hspace{1cm} (A13)

The vector of unknowns is the same as in the case of Appendix B:

\[ p = (a_0, a_1, a_2, a_3, b_0, b_1, b_2, b_3, c_0, c_1, c_2, c_3)^T \]  \hspace{1cm} (A14)

In the Jacobian matrix \( A \), the \( r_1 \) parameter is omitted and also the reduced observation vector \( l \) differs:

\[ A = (A_1 \ A_2 \ A_3) \]  \hspace{1cm} (A15)

with the submatrices:

\[ A_1 = \begin{pmatrix} \frac{\partial f}{\partial x_0} & \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \frac{\partial f}{\partial x_3} & \frac{\partial f}{\partial b_0} & \frac{\partial f}{\partial b_1} & \frac{\partial f}{\partial b_2} & \frac{\partial f}{\partial b_3} & \frac{\partial f}{\partial c_0} & \frac{\partial f}{\partial c_1} & \frac{\partial f}{\partial c_2} & \frac{\partial f}{\partial c_3} \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix} \]  \hspace{1cm} (A16)

\[ A_2 = \begin{pmatrix} \frac{\partial f}{\partial x_0} & \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \frac{\partial f}{\partial x_3} & \frac{\partial f}{\partial y_0} & \frac{\partial f}{\partial y_1} & \frac{\partial f}{\partial y_2} & \frac{\partial f}{\partial y_3} & \frac{\partial f}{\partial g_0} & \frac{\partial f}{\partial g_1} & \frac{\partial f}{\partial g_2} & \frac{\partial f}{\partial g_3} \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix} \]  \hspace{1cm} (A17)

\[ A_3 = \begin{pmatrix} \frac{\partial f}{\partial x_0} & \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \frac{\partial f}{\partial x_3} & \frac{\partial f}{\partial z_0} & \frac{\partial f}{\partial z_1} & \frac{\partial f}{\partial z_2} & \frac{\partial f}{\partial z_3} & \frac{\partial f}{\partial h_0} & \frac{\partial f}{\partial h_1} & \frac{\partial f}{\partial h_2} & \frac{\partial f}{\partial h_3} \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix} \]  \hspace{1cm} (A18)

\[ l = \begin{pmatrix} f_1 - g_1 \\ f_2 - g_2 \\ \vdots \\ f_n - g_n \end{pmatrix} \]  \hspace{1cm} (A19)

The normal matrix \( A^T \cdot A \) (with the \( G_{kl,i} \) matrices from Equations (27) and (30)) is:
Appendix C.2. Computational Effort

Appendix C.2.1. Direct Method

Similar to the case above, for each voxel, the following terms are computed and saved in temporal variables because they are used several times:

\[
g_{x,i} = g_{x,i} \cdot \bar{x}_i \\
g_{y,i} = g_{y,i} \cdot \bar{y}_i \\
g_{z,i} = g_{z,i} \cdot \bar{z}_i
\]  

(A22)

60 terms of the matrices \(G_{11}, G_{12}, G_{13}, G_{22}, G_{23}\) and \(G_{33}\) (Equation (30)) and 12 terms of \(A^T \cdot 1\) are computed. To evaluate the computational effort of the direct method, the number of additions and multiplications are counted. Only the operations per voxel are considered, the other operations can be neglected.

- Multiplications per voxel:
  - Nine multiplications for the temporally terms of Equation (A22).
  - 60 multiplications in the \(G_{ij}\) matrices.
  - 12 multiplications in the sums of \(A^T \cdot 1\).

- Additions per voxel:
  - One addition (subtraction) in \(df_i = f_i - g_i\).
  - 60 additions in the 60 sums of the \(G_{ij}\) matrices.
  - 12 additions in the 12 sums of \(A^T \cdot 1\).

All in all, 81 multiplications and 73 additions have to be computed per voxel for the direct method.

Appendix C.2.2. Standard Method

For the standard method, the buildup of \(A\) and \(I\) as well as the matrix multiplications are considered:

- Multiplications per voxel:
Nine multiplications for building up the Jacobian matrix of Equation (A15).
- 78 multiplications (78 elements) due to the matrix multiplication of $A^T \cdot A$ (only upper triangular matrix).
- 12 multiplications (12 elements) due to the matrix-vector multiplication of $A^T \cdot l$.

**Additions per voxel:**
- One addition (subtraction) for building up the reduced observation vector $l$.
- 78 additions (78 elements) due to the matrix multiplication of $A^T \cdot A$ (only upper triangular matrix).
- 12 additions (12 elements) due to the matrix-vector multiplication of $A^T \cdot l$.

Thus, for the standard method, 99 multiplications and 91 additions have to be computed per voxel.

Appendix C.2.3. Comparison

Table A2 shows the number of computations per voxel for the direct and the standard method. Using the direct method, 18.2% less multiplications and 19.8% less additions are done.

**Table A2.** Comparison of the computational effort between the direct computation of the normal equations and the standard method (affine mathematical model without radiometric parameters).

| Method     | Multiplications per vx | Additions per vx |
|------------|------------------------|------------------|
| Standard   | 99                     | 91               |
| Direct     | 81                     | 73               |

**References**

1. Barnea, D.I.; Silverman, H.F. A Class of Algorithms for Fast Digital Image Registration. *IEEE Trans. Comput.* 1972, C-21, 179–186. [CrossRef]
2. Lucas, B.D.; Kanade, T. An Iterative Image Registration Technique with an Application to Stereo Vision. In Proceedings of the 7th International Joint Conference on Artificial Intelligence, Vancouver, BC, Canada, 24–28 August 1981; Volume 2, pp. 679–679. [CrossRef]
3. Ackermann, F. Digital Image Correlation: Performance and Potential Application in Photogrammetry. *Photogramm. Rec.* 1984, 11, 429–439. [CrossRef]
4. Grün, A. Adaptive least squares correlation: A powerful image matching technique. *S. Afr. J. Photogramm. Remote Sens. Cartogr.* 1985, 14, 175–187.
5. Bethmann, F.; Luhmann, T. Least-squares matching with advanced geometric transformation models. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* 2010, 38 Pt 5, 86–91.
6. Grün, A.; Baltzavias, E.P. Geometrically constrained multiphoto matching. *Photogramm. Eng. Remote Sens.* 1988, 54, 633–641.
7. Wrobel, B.P. Digital image matching by facets using object space models. In *Advances in Image Processing*; Oosterlinck, A.J., Tescher, A.G., Eds.; International Society for Optics and Photonics (SPIE): Bellingham, WA, USA, 1987; Volume 804, pp. 325–335. [CrossRef]
8. Ebner, H.; Heipke, C. Integration of digital image matching and object surface reconstruction. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* 1988, XXVII, 534–545.
9. Maas, H.G.; Stefanidis, A.; Grün, A. From pixels to voxels: Tracking volume elements in sequences of 3D digital images. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* 1994, 2357, 539–546. [CrossRef]
10. Bay, B.K.; Smith, T.S.; Fyhrie, D.P.; Saad, M. Digital Volume Correlation: Three-dimensional Strain Mapping Using X-ray Tomography. *Exp. Mech.* 1999, 39, 217–226. [CrossRef]
11. Liebold, F.; Lorenzonzi, R.; Curosu, I.; Léonard, F.; Mechtcherine, V.; Paciornik, S.; Maas, H.G. 3D least squares matching applied to micro-tomography data. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* 2021, 43, 533–539. [CrossRef]
12. Tudisco, E.; Andò, E.; Cailletaud, R.; Hall, S.A. TomoWarp2: A local digital volume correlation code. *SoftwareX* 2017, 6, 267–270. [CrossRef]
13. Stamati, O.; Andò, E.; Roubin, E.; Cailletaud, R.; Wiebicke, M.; Pinzon, G.; Couture, C.; Hurley, R.C.; Cauk, R.; Caillerie, D.; et al. spam: Software for Practical Analysis of Materials. *J. Open Source Softw.* 2020, 5, 2286. [CrossRef]
14. Yang, J.; Hazlett, L.; Landauer, A.K.; Franck, C. Augmented Lagrangian Digital Volume Correlation (ALDVC). *Exp. Mech.* 2020, 60, 1205–1223. [CrossRef]
15. Sutton, M.A.; Orteu, J.J.; Schreier, H. *Image Correlation for Shape, Motion and Deformation Measurements: Basic Concepts, Theory and Applications*, 1st ed.; Springer: Berlin/Heidelberg, Germany, 2009. [CrossRef]
16. Jin, H.; Favaro, P.; Soatto, S. Real-Time Feature Tracking and Outlier Rejection with Changes in Illumination. In Proceedings of the Eighth IEEE International Conference on Computer Vision (ICCV), Vancouver, BC, Canada, 7–14 July 2001; Volume 1, pp. 684–689. [CrossRef]

17. Keys, R. Cubic Convolution Interpolation for Digital Image Processing. *IEEE Trans. Acoust. Speech Signal Process.* 1981, 29, 1153–1160. [CrossRef]

18. Lorenzoni, R.; Curosu, I.; Léonard, F.; Paciornik, S.; Mechtcherine, V.; Silva, F.A.; Bruno, G. Combined mechanical and 3D-microstructural analysis of strain-hardening cement-based composites (SHCC) by in-situ X-ray microtomography. *Cem. Concr. Res.* 2020, 136, 106139. [CrossRef]

19. Lewis, J.P. Fast Template Matching. In Proceedings of the Vision Interface Conference, Quebec City, QC, Canada, 15–19 May 1995; Volume 95, pp. 120–123.

20. Urschler, M.; Bornik, A.; Donoser, M. Memory Efficient 3D Integral Volumes. In Proceedings of the 2013 IEEE International Conference on Computer Vision Workshops, Sydney, Australia, 2–8 December 2013; pp. 722–729. [CrossRef]