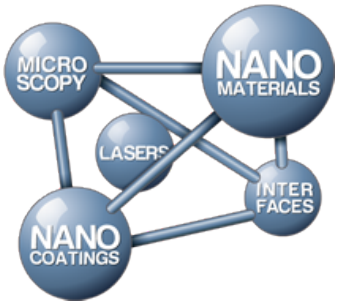
university of groningen / faculty of mathematics and natural sciences / zernike institute for advanced materials

Using digital image correlation for 3D surface topography in scanning electron microscopy

Joost Tijmen Ouwerling

s2176378 j.t.ouwerling@student.rug.nl

April - June 2015



Bachelor Research Thesis

Materials Science Group

University of Groningen

Supervisor:	dr. ir. E.T. Faber
First examiner:	prof. dr. Jeff Th.M. de Hosson
Second examiner:	dr. V. Ocelìk

Summary

Digital Image Correlation (DIC) can be used to approximate 3D surface topography of SEM images using a geometrical approach. [1] This report discusses a literature study and implementation in Matlab of a DIC algorithm, which has to replace the current system for it. Two so-called correlation coefficients, a measurement of how well a certain displacement maps objects correctly between two images, are analyzed, namely the Cross Correlation Coefficient (CC) and the Sum of Squared Differences (SSD). After normalization and a correction for the offset in brightness, the SSD coefficient proved to be the best choice due to its computational efficiency. Three different DIC algorithms, Coarse Fine search, Forward Additive Newton Rhapson (FA-NR) and Inverse Compositional Gauss Newton (IC-GN) were compared. IC-GN turned out to be the most acuurate and efficient, in terms of computing time. An implementation in Matlab of the above algorithm, using a reliability guided displacement tracking strategy and bicubic interpolation, is discussed. At the end, several considerations and possible optimizations are discussed.

Acknowledgment

I would like to thank the following persons for their great help during this project. First of all, dr. ir. E.T. (Enne) Faber, for his excellent daily supervision and inspirational informal talks, which kept me going throughout the project. Secondly, prof. dr. J. Th. M. (Jeff) de Hosson, for having me in his research group and providing valuable feedback. Also, dr. V. (Vaclav) Ocelik, for being my second examiner.

Joost Tijmen Ouwerling

June 2015

Contents

Chapter 1

Introduction

This chapter gives an introduction to the background and the objectives of the research. Furthermore, the approach of the research and structure of this report is elaborated.

1.1 Background

In an article by Faber et al [1] a new method for obtaining reliable 3D surface topography reconstruction from 2D Scanning Electron Microscopy (SEM) images is presented. This method uses three images, taken at different tilt angles, to reconstruct the topography of the sample. The strength of this method lies in the fact that the precise knowledge about the tilt does not need to be known, thereby eliminating the need for complex calibrations of the SEM. Convential 2D Digital Image Correlation (DIC) is used to obtain the displacement δ of every point in the images after tilting. By evaluating these displacements, the z coordinates of the sample can be reconstructed. This new method is different from the more traditional way of obtaining 3D topography, since it *derives* the rotation parameters from the images, while traditional methods *depend* on accurate calibration and precise control of the rotation.

A schematic representation of the tilting is shown in figure 1.1. A SEM image can be considered as a matrix of greyscale values $\{x_i,\ y_i\}$. The initial situation is labeled A and the situation after tilting is labeled B. The coordinates, after observation, of A and B are compared using DIC which results in the displacements $\{dx_{i,AB},\ dy_{i,AB}\}$.

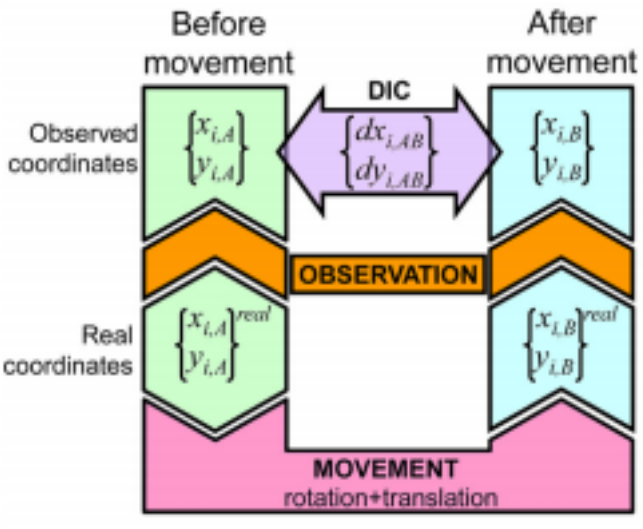


Figure 1.1: The tilting process of two images [1]

Currently, the DIC process is done by the Aramis 5.3 software, which is designed for optically acquired images. This seems to give the right displacements, but has some disadvantages. Most important, there is no error estimation given for the displacements, making it less useful (or even unuseful) for scientific research and engineering applications. Also, the code can not be viewed or modified, leaving the inner workings of the process completely obscured. Lastly, the Aramis does not take SEM specific characteristics into account. Since scientists need to know what exactly is happening and how precise the experiment results are, so they can make valid assumptions, the Aramis 5.3 is an undesirable dependency in the process.

1.2 Objectives

The main objective for this research is to set up a software system which can do DIC on SEM generated images. The input of the program are two SEM images, in which every pixel contains a greyscale value. The output of the program should be a displacement vector which maps every pixel / physical point in the first image to the same physical point (probably at a different pixel location) in the second image, including an error estimation. It is important that all assumptions and limitations of the implementations are clearly documented. Also, the implementation should be easy to extend and well documented, so it can be used in further studies.

1.3 SEM specific considerations (to end!)

Initially, DIC algorithms for optically acquired images are promising as a starting point. One would guess that these are already well known in (computing) science, considering its wide range of applications in for example object recognition. However, there are many characteristics of SEM acquired images which differ significantly from optical ones. A more advanced and therefore more useful algorithm would take these characteristics into account. Some of these characteristics include:

- **The scanning nature of SEM**
  Optical camera's create an image in one instant, by capturing the whole visible region at a certain time. SEM images of size $n \times m$ are acquired by a scanning electron beam which traverses from $\{x, y\} = \{0, 0\}$ to $\{x, y\} = \{n, m\}$ in a certain time range. Within this time range, the sample could move a little bit or environmental influences (i.e. vibrations) could change the scanning parameters while taking an image.

- **The sample could get electrostatically charged** [2, p. 14]
  If a sample has no electrical conductivity, absorbed electrons from the electron beam can charge the sample. This causes many errors in observations. Normally, metal coatings or low pressure observations are used to prevent charging, but slight charging happens quickly and should be taken into account.

- **SEM specific noise**
  Different sources can generate noise on the SEM acquired images as compared to optical images. This noise has to be identified and not taken into account in the DIC process. Of course, this noise is kept as low as possible during the image capturing, but will appear nonetheless. A well designed SEM optimized DIC algorithm could filter out these noises and thereby give a better result.

1.4 Approach

The approach in this project consists out of five global steps. The most important target is to get insight in DIC algorithms and implement one, or use an existing open-source solution. Later on, if time permits, improvements can be added in a continous process.

1. **Literature Study**
   The first step in the research will be a literature study on DIC, since we have to know which

algorithms are available and are the most applicable for our goals. This will also benefit our implementation process, since we won't run into problems that occured to other people before.

2. **Implementation Decision**
   After the literature study, it should be investigated whether or not there exists an open source library which servers our needs. If this does not exist or if it requires a lot of effort to implement, a home-made DIC algorithm will be implemented.

3. **Implementation**
   Based on the decision made above, the DIC will be implemented using an existing open source library or a home-made program. The implementation should adhere to the requirements of the objective.

4. **Verification**
   Using known samples, the topography reconstruction method including the implemented DIC process should be tested using actual SEM measurements. This is a necessary step in the project to ensure a correctly working program.

5. **Continuous improvements**
   Since it is not known beforehand how difficult the implementation and SEM specific considerations are, after setting up an initial DIC algorithm continuous improvements can be made to the algorithm. Each of these improvements will involve the steps 1 to 4.

1.5 Structure of the report

The report has three important sections. First of all, the theory of current DIC algorithms is discussed in chapter 3. In chapter 4, an implementation in Matlab is discussed. After that, several considerations and possible improvements are discussed in chapter 5.

Chapter 2

Key concepts of Scanning Electron Microscopy

This chapter serves as a concise introduction to the key concepts in SEM which are relevant for this research. Readers who are familiar with the workings of the SEM can skip this section and continue in the next chapter.

2.1 Introduction

The fundamental principles of SEM were discovered around the 1930s in Germany. After World War II, this research was continued in the U.K. and in 1965 the first commercial SEM was accounced. Since then, a lot has improved and nowadays SEM is an indispensible tool in materials science and other scientific fields. The resolution range is in the order of 10 nm and below, making it an order of magnitude more than an optical microscope. Useful magnifications up to 150,000 are possible. There is a wide range of applications for the SEM. Examples are compositional observation of the surface, fracture research, corrosion and wear studies. This research focusses on another application, namely the topography of the sample.

2.2 Working principles

The SEM works by letting an electron beam irradiate on the surface of a sample, along closely spaced lines, like a cathode ray tube used for imaging on televisions. Interactions between the beam and the atoms in the sample produce various kind of information in the form of different sorts of electrons, x-rays and electromotive force, as shown in figure 2.1. How exactly the

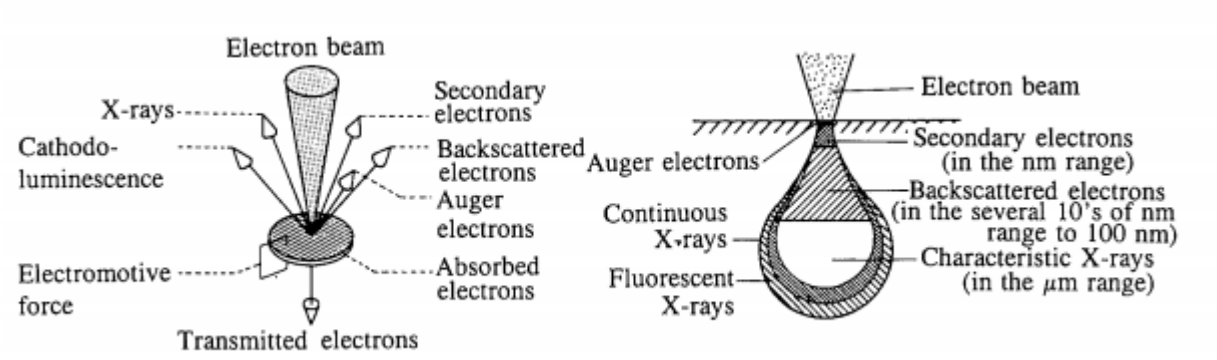


Figure 2.1: The left figure shows the different sources of information. The figure on the right information shows where in the sample these information sources originate. [2]

electron beam is generated and how the different sources of information are detected is outside the scope of this introduction, but if the reader is interested, [2] and [3] are good sources of information.

The secondary electrons (SE) are the most interesting for this research. They are formed by the interaction between the incoming electrons with the loosely bound atomic electrons. Their energy is in the range of approximately 1 to 50 eV, with a maximum frequency at 3 eV. Over half of the emitted SE originate within a depth of 0.5 nm, making the depth of information approximately 1 - 10 nm [3]. Therefore, the number of SE emitted from the sample greatly depend on the incident angle of the electron beam on the sample. This is the reason why SE generally provide information about the topography of the sample.

The SE are collected by a scintillator. Since the SE have very low energy, a voltage of approx. 10 kV applied to the scintillator will collect all of the emitted SE, even if they were emitted in the other direction. The number of collected SE are transformed to an electronic signal, which is, after amplification, displayed on a screen. When an image is captured, the beam scans a certain region of the sample and outputs the relative difference in the number of collected SE, in the form of a greyscale image.

Chapter 3

An introduction to Digital Image Correlation

Digital Image Correlatuion (DIC) is a technique to measure displacements of physical objects from a reference image into a target image. The basic structure of a DIC algorithm is to optimize a certain correlation coefficient, which indicates how well a certain displacement field maps points from the reference image to the target image. The algorithm calculates this coefficient based on the natural surface texture or on an artificial speckle pattern applied to the sample. DIC is widely used in science and engineering due to its advantages of experimental setup and sample preparation. The best known application is the measurements of material deformation, in which the displacements are calculated after, for example, a force is applied to a material. However, it is a general technique and is not restricted to its uses in experimental mechanics. Other applications include the optical mouse and military target recognition. The technique was used as early as 1975, but much of the early work in mechanics has been done at the University of South Carolina at the early 1980's [4] [5].

3.1 Mathematical definition

The DIC algorithm takes two greyscale images as an input, in which every pixel represents the intensity at the (x, y) coordinate. Let $f(x, y)$ and $g(x, y)$ be the greyscale values at a specified (x, y) pixel for respectively the reference and target image. We want to find a displacement field $\mathbf{D}$, which is defined as

$$\mathbf{D}(x, y) = u(x, y)\mathbf{x} + v(x, y)\mathbf{y} \tag{3.1}$$

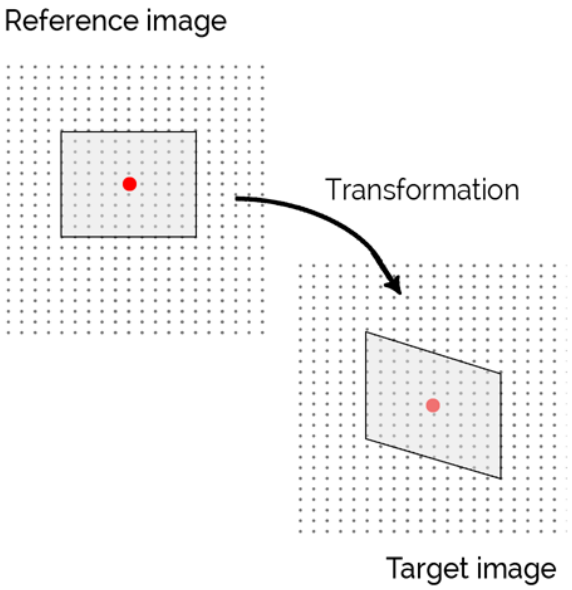


Figure 3.1: A visual definition of digital image correlation

which maps every object in $f(x, y)$ correctly to $g(x', y')$ in which x' and y' are defined as

$$x' = x + u(x, y) \tag{3.2}$$

$$y' = y + v(x, y) \tag{3.3}$$

This means that we have to find the vector components u and v of **D** for which a certain correlation factor C is a global optimum, since the displacement field corresponding to the optimal correlation factor is the best approximation which can be made. This transformation has been made visual in figure 3.1.

3.2 Geometrical definitions

DIC will be performed on several gridpoints on the reference image, which can (and probably will) all have a different displacement. These gridpoints are separated by a certain number of pixels in the x and y direction, which represent columns and rows in matrix terms. This is exemplified in figure 3.2, where the red pixels are gridpoints.

The correlation match is based on a subset, which is a certain area around the gridpoints which are matched with the same area in the target image. That is, for every pixel in the reference subset, the displacement field **D** is applied to it. This corresponds to the grey area in figure 3.1 and is further elaborated in figure 3.3.

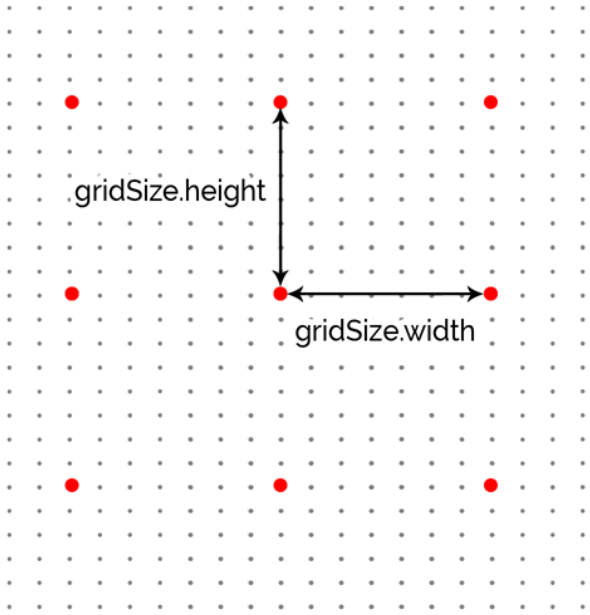


Figure 3.2: A visual definition of the gridpoints. The grey points are regular pixels, the larger red points are gridpoints. The gridsize holds information about the spacing between different gridpoints.

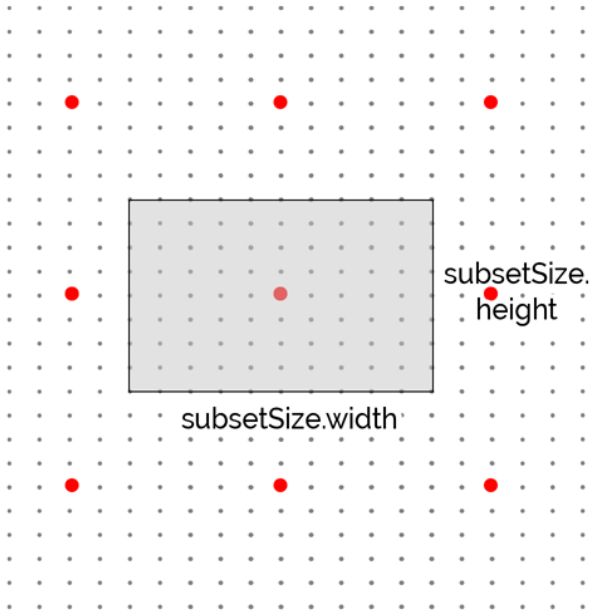


Figure 3.3: A visual definition of the subset. The color definitions are the same as in figure 3.2. The grey area is the subset.

3.3 Correlation criteria

First a little note about the mathematical notation in he equations in this section. The summations are, without explicitly stating it, over all (x_i, y_j) coordinates in the region of interest. Also, to improve readability, f_{ij} is defined as $f(x_i, y_j)$ and g_{ij} as $g(x'_i, y'_j)$.

Within the literature, different correlation factors are used [5][6]. As this coefficient directly influences the resulting displacement field, the coefficient used in the algorithm is a significant choice. The most intuitive - in my opinion - is the so-called *Sum of Squared Differences* (SSD) criterion and it needs to be minimized to obtain a maximum correlation. It is defined, in its most basic form, as

$$C_{SSD} = \sum \left[f_{ij} - g_{ij} \right]^2 \tag{3.4}$$

Note the fact that we use a summation instead of an area integral. This is because the input is discrete in nature. To take into account the offset in intensity that can exist between the two images, the mean value of the intensity can be subtracted,

$$C_{ZSSD} = \sum \left[\left(f_{ij} - f_m \right) - \left(g_{ij} - g_m \right) \right]^2 \tag{3.5}$$

in which $g_m = \frac{1}{n} \sum g_{ij}$ and $f_m = \frac{1}{n} \sum f_{ij}$, where n is the number of analyzed pixels in subset. The necessity for this correction can be seen in figure 3.4, in which two SEM images of the same sample are shown. You can clearly see the offset in brightness between the two images.

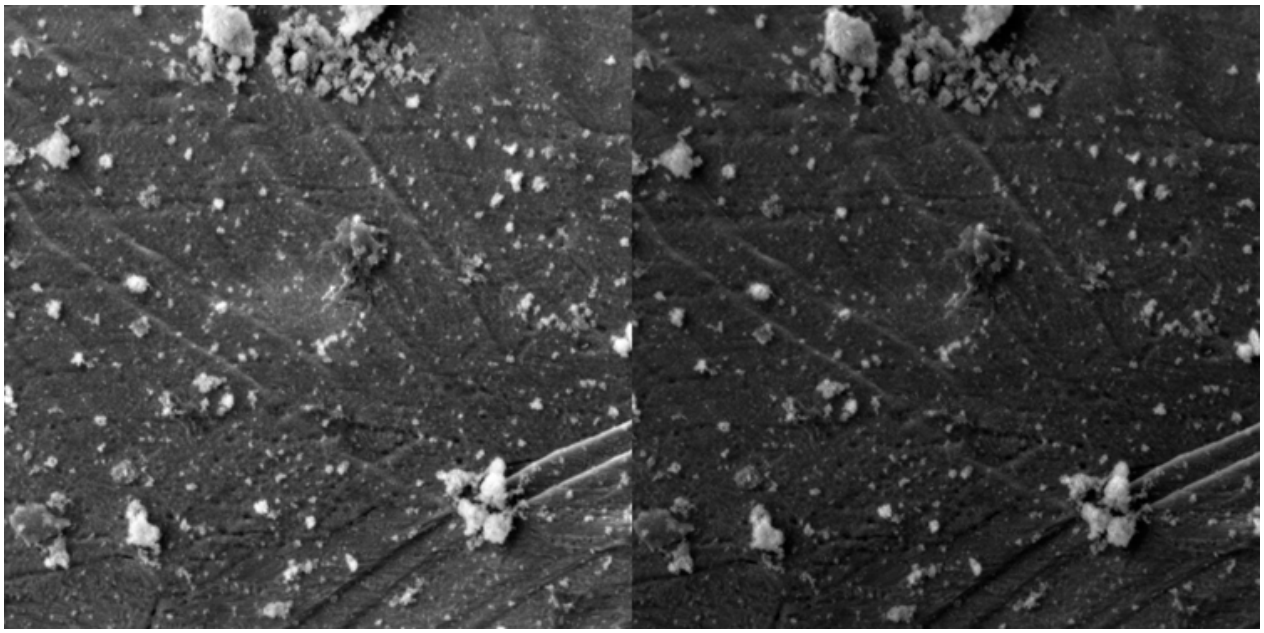


Figure 3.4: Two different SEM images of the same sample with a visible difference in brightness.

The influence of the scale change of the intensity between the two images can be removed by normalizing the correlation coefficient. This gives a final *Zero mean Normalized Sum of Squared Differences* (ZNSSD) correlation factor of

$$C_{ZNSSD} = \sum \left(\frac{\bar{f}_{ij}}{\sqrt{\sum \bar{f}_{ij}^2}} - \frac{\bar{g}_{ij}}{\sqrt{\sum \bar{g}_{ij}^2}} \right)^2 \tag{3.6}$$

in which $\bar{g}_{ij} = g_{ij} - g_m$ and $\bar{f}_{ij} = f_{ij} - f_m$.

Another recurring correlation coefficient in the literature is the so-called direct cross correlation factor (CC). This factor needs to be maximized and is writte, in its simplest form, as

$$C_{CC} = \sum f_{ij} g_{ij} \tag{3.7}$$

Again, the intensity offset and scale change can be accounted for by substracting the mean and normalizing the function. This results in the *Zero mean Normalized Cross-Correlation* coefficient, defined as

$$C_{ZNCC} = \frac{\sum \bar{f}_{ij} \bar{g}_{ij}}{\sqrt{\sum \bar{f}_{ij}^2 \sum \bar{g}_{ij}^2}} \tag{3.8}$$

One can prove algebraically [7] that C_{ZNSSD} and C_{ZNCC} are related by

$$C_{ZNSSD} = 2(1 - C_{ZNCC}) \tag{3.9}$$

which means that the resulting displacement field won't differ for these two correlation coefficients. There is no clear advantage for either of the coefficients. The optimization of C_{ZNSSD} is more easy and is therefore used a lot in practice. However, in certain DIC approached C_{ZNCC} is more favorable since it intuitevely expresses the reliability of the match, taking its range ([-1,1] for C_{ZNCC} vs. [0,4] for C_{ZNSSD}) into account.

A few other criteria can be found in the literature. For a more extensive analysis of these, including C_{ZNCC} and C_{ZNSSD}, the reader is referred to [7]. In this paper, the different correlation criteria are derived and it is shown by a theoretical analysis that they are equivalent. This is verified by numerical simulationa as well as an actual experiment.

3.4 General algorithm

Generally, to find the complete displacement field **D**, DIC algorithms analyze small subsets of the images, of which the combined results form the complete field. The reference image is partitioned in smaller regions, referred to as subsets, in which the displacement field is assumed to be homogeneous. Let (x_0, y_0) be the center of the subset, and let S be all coordinates within the subset. If S is small enough, we can do a linear approximation of the displacement field using a first-order Taylor expansion around (x_0, y_0). The values for x' and y' then become

$$x' = x + u + \frac{\partial u}{\partial x}(x - x_0) + \frac{\partial u}{\partial y}(y - y_0) \tag{3.10}$$

$$y' = y + v + \frac{\partial v}{\partial x}(x - x_0) + \frac{\partial v}{\partial y}(y - y_0) \tag{3.11}$$

Note again that, since the displacement field is homogeneous within the subset, the displacement u, v and their derivatives are constant in S. The six unknowns in the displacement field can be grouped together in a vector **p**

$$\mathbf{p} = \begin{bmatrix} u & v & \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix}^T \tag{3.12}$$

3.5 Optimization strategies

The problem we have left is to find the best approximation of **p**. In the next paragraphs, three methods are discussed: Coarse-fine search, Forward Additive Newton Rhapson (FA-NR) and Inverse Compositional Gauss Newton (IC-GN).

3.5.1 Coarse fine search

Coarse fine search is the most obvious way to find the optimal **p** and it has the characteristics of a brute force algorihtm. By simply trying all the possible combinations of **p** and comparing their correlation coefficient, the best approximation can be found. However, this is very computing intensive, up to a point where it is not viable to use anymore. There exist improvements for this technique. First of all, the so-called nested search approach can be used. With this approach, the search region decreases while the search step size also decreases, resulting in an increasing search accuracy in a decreasing area, leaving the computational cost the same for every step. Another optimization can be to search for the best correlation coefficients in pairs of values of **p**. First, look for the best u and v. Then, look for the best $\frac{\partial u}{\partial x}$ and $\frac{\partial v}{\partial y}$. Then, look for the best $\frac{\partial u}{\partial y}$ and $\frac{\partial v}{\partial x}$. If you keep continuing this iteration until the algorithm converges, you will find the best approximation faster than trying all six parameters at the same time.

Even with these optimizations, this approach is still somewhat *dumb*. We can approach this problem also from a mathematical point of view, giving a more accurate, fast and in a certain sense more beautiful algorithms. That is what is done in the next two methods. This does not mean that the coarse-fine search has to be thrown overboard completely. It can still be used to estimate a rough initial guess, which can used as a starting point in more sophisticated algorithms.

3.5.2 Forward Additive Newton Rhapson

Let's step back for a minute and think about what we actually want to achieve. We have a function, the correlation factor, which takes **p** as a parameter. We need to minimize the value of this function. In elementary, one dimensional calculus, this can be achieved by taking the derivative and equate it to zero. It turns out, under certain conditions, that this also holds for multiple dimensions. Thus, we need to solve the following set of six equations:

$$\frac{\partial C}{\partial \mathbf{p}} = \mathbf{0} \quad \text{,or} \quad C_{p_i} = 0 \tag{3.13}$$

In numerical mathematics, the Newton Rhapson method is often used for finding the root of a function. This method can be extended to find the root of multiple simultaneous equations, for example, when finding the six roots of our problem. This method has been introduced by Bruck et al [5]. We can derive the necessary improvement iteration as follows. Consider the Taylor

series of one of our functions:

$$C_{p_i}(\mathbf{p}+\delta\mathbf{p}) = C_{p_i}(\mathbf{p}) + \sum_j \frac{\partial C_{p_i}}{\partial p_j}\delta p_j + O(\delta\mathbf{p}^2) \tag{3.14}$$

Let's assume that $\delta\mathbf{p}$ is small, so we can drop the $\delta\mathbf{x}^2$ (and higher order) term at the end. Let $C_{p_i}(\mathbf{p}+\delta\mathbf{p}) = 0$, i.e. approximate it as the final answer. This results in the following equation:

$$0 = C_{p_i}(\mathbf{p}) + \sum_j \frac{\partial C_{p_i}}{\partial p_j}\delta p_j \tag{3.15}$$

The matrix of partial derivatives in the summation is the Jacobian of C_{p_i}, but since C_{p_i} are already partial derivatives, this is the Hessian of C:

$$\mathbf{H}_{ij} = \frac{\partial C_{p_i}}{\partial p_j} = \frac{\partial^2 C}{\partial p_i \partial p_j} \tag{3.16}$$

Now, together with the definition of the Hessian, after solving equation 3.15 for $\delta\mathbf{p}$,

$$\delta\mathbf{p} = -\mathbf{H}^{-1}(\mathbf{p})C_{p_i}(\mathbf{p}) \tag{3.17}$$

we get the iteration step,

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \delta\mathbf{p} = \mathbf{p}_k - \mathbf{H}^{-1}(\mathbf{p}_k)C_{\mathbf{p}}(\mathbf{p}_k) \tag{3.18}$$

which has to be executed until the solution has converged or a certain number of iteration steps has been computed. A schematic overview of the process is shown in figure 3.5.

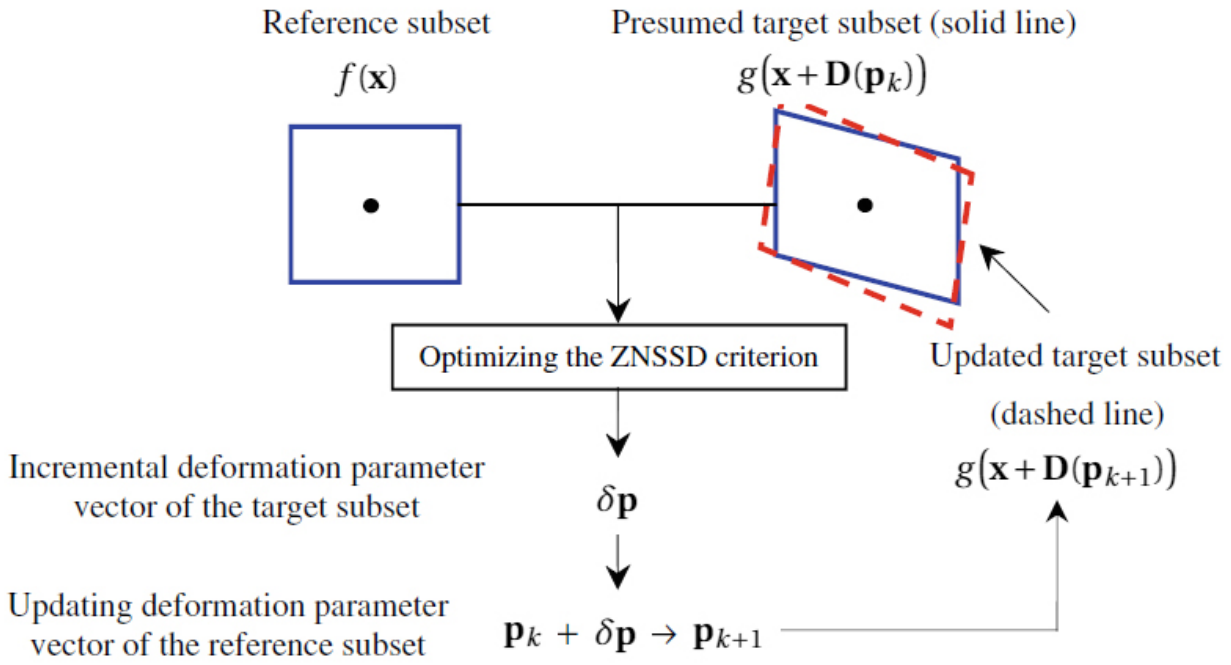


Figure 3.5: A schematic process overview for the FA-NR algorithm [8]. The blue square is the current situation, the red, dashed block after an iteration.

Constraints and considerations

The method as described above has some considerations and constraints which needs to be taken into account. First of all, it is possible that this method finds a local minimum instead of an absolute minimum, which would result in a false result. There is, unfortunately, no way to distinguish what kind of minimum we have found. To be sure to find the absolute minimum, the initial guess needs to be close enough to the true answer. It is difficult to quantitize this 'closeness', but according to [5], the coarse-fine search with an accuracy of one pixel gave an adequate guess in almost all cases. Secondly, we have dropped higher order terms from our Taylor series. This is an approximation we can do if we have, like above, a close initial guess. $\delta\mathbf{p}$ will then be small, making the higher order terms negligible. There are also several numerical considerations in implementing this algorithm, but I leave those out for now.

3.5.3 Inverse Compositional Gauss Newton

The FA-NR algorithm produces quite accurate but requires a lot of computation time - the Hessian has to be recomputed in every iteration. There exists another approach for image align-

ment, the compositional approach, which uses an iterative process to solve for an incremental warp rather than an additive update to $\mathbf{p}$. It can be proved that these approaches are equivalent [9]. For both algorithms there exist an inverse variant, in which the role of the target and reference image is reversed, to prevent recalculating the hessian in every iteration. However, the inverse variant of the additive algorithm works only for a small subset of warps. The *inverse compositional* method can handle a large collection of warps and is almost as efficient as the inverse additive algorithm. Therefore, the inverse compositional algorithm is what we'll be using in our program. For an excellent discussion on all of the above, I'd like to refer to [9].

Pan et al [8] implemented this inverse compositional algorithm specifically for digital image correlation. Since we are solving a least squares problem, the numerical *Gauss Newton* algorithm is used, which immediately explains the name. The *warp function* $\mathbf{W}$ for our problem is as follows

$$\mathbf{W}(\mathbf{x},\mathbf{p}) = \begin{pmatrix} 1+u_x & u_y & u \\ v_x & 1+v_y & v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.19}$$

where x and y are the positions relative to the subset's center. We can convert this warp function into our The ZNSSD criterion, equation 3.6, then becomes

$$C_{ZNSSD} = \sum \left(\frac{f(\mathbf{x}) - \bar{f}}{\sqrt{\sum \bar{f}_{ij}^2}} - \frac{g(\mathbf{x}+\mathbf{W}(\mathbf{x},\mathbf{p})) - \bar{g}}{\sqrt{\sum \bar{g}_{ij}^2}} \right)^2 \tag{3.20}$$

This is nothing new - just a different notation. In this notation, the FA-NR algorithm tries to optimize the coefficient by finding the optimal value $\delta\mathbf{p}$, by minimizing the correlation coefficient for $\mathbf{W}(\mathbf{x},\mathbf{p}) = \mathbf{W}(\mathbf{x},\mathbf{p}+\delta\mathbf{p})$. Instead of this additive process, the IC-GN method reverses the roles of the target and reference images. The following equation is minimized,

$$C_{ZNSSD} = \sum \left(\frac{f(\mathbf{x}+\mathbf{W}(\mathbf{x},\delta\mathbf{p})) - \bar{f}}{\sqrt{\sum \bar{f}_{ij}^2}} - \frac{g(\mathbf{x}+\mathbf{W}(\mathbf{x},\mathbf{p})) - \bar{g}}{\sqrt{\sum \bar{g}_{ij}^2}} \right)^2 \tag{3.21}$$

after which the optimal $\delta\mathbf{p}$ is merged into $\mathbf{p}$ in the following way

$$\mathbf{W}(\mathbf{x},\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x},\mathbf{p}) \circ \mathbf{W}(\mathbf{x},\delta\mathbf{p})^{-1} \tag{3.22}$$

First, we have to find a way to determine the minimum of equation 3.21 with respect to $\delta\mathbf{p}$. The derivation of this equation can be found in Appendix A.1 and the result is

$$\delta\mathbf{p} = -\mathbf{H}^{-1} \times \sum \left\{ \left(\nabla f \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right)^T \left(\left(f(\mathbf{x}) - \bar{f} \right) - \frac{\Delta f}{\Delta g} \left(g(\mathbf{x} + \mathbf{W}(\mathbf{x}, \mathbf{p})) - \bar{g} \right) \right) \right\} \tag{3.23}$$

in which

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) \tag{3.24}$$

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{pmatrix} 1 & \Delta x & \Delta y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta x & \Delta y \end{pmatrix} \tag{3.25}$$

and $\mathbf{H}$ is the 6×6 Hessian matrix defined as

$$\mathbf{H} = \sum \left\{ \left(\nabla f \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right)^T \left(\nabla f \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right) \right\} \tag{3.26}$$

The main computational advantage of the IC-GN method over the FA-NR method lies in the definition of this Hessian. If you take a careful look at it, you can see that there is no dependence on $\mathbf{p}$ or $\delta\mathbf{p}$ in the Hessian. This means that it can be precomputed, whereas the Hessian needs to be recalculated in every iteration with the FA-NR algorithm. Also, IC-GN only needs the intensity at sub-pixel locations of the target image g, whereas the FA-NR algorithm needs the gradients of g as well. A schematic overview of the IC-GN method is shown in figure 3.6.

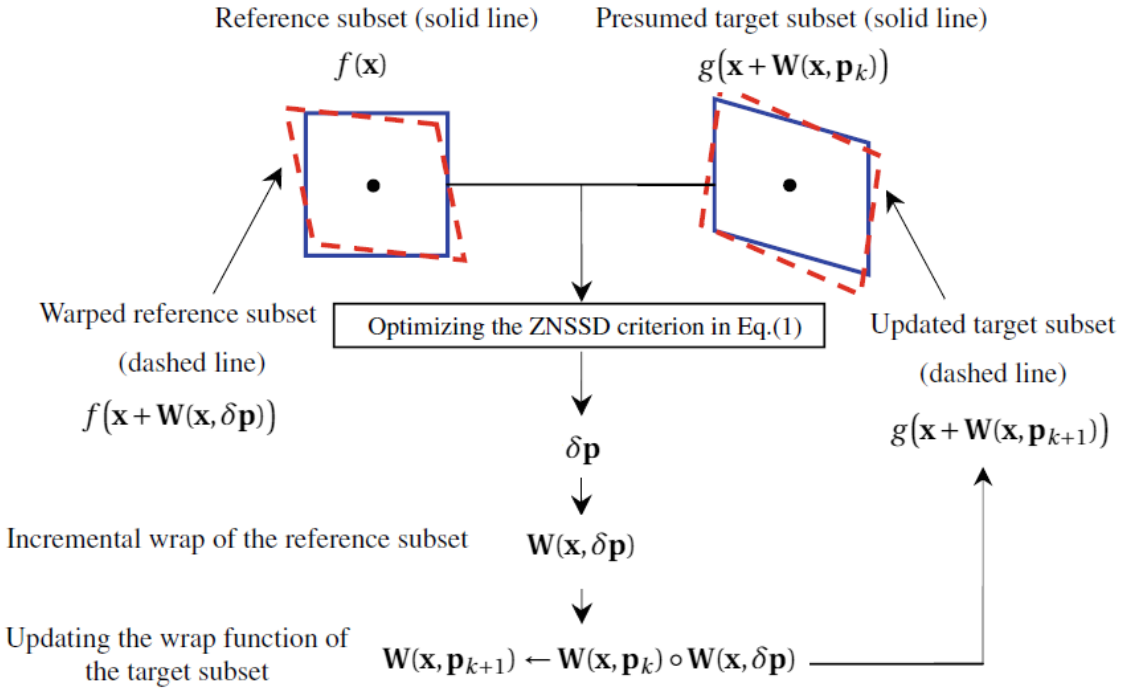


Figure 3.6: A schematic process overview for the IC-GN algorithm [8]. The blue square is the current situation, the red, dashed block after an iteration.

3.6 Interpolation

As stated in the previous section, IC-GN (FA-NR as well) depend on subpixel intensity as well. To acquire these values, we have to interpolate our dataset. Since the interpolation directly influences our end result, choosing the right scheme is an important choice. There are several algorithms available, and for 2D data sets, bilinear interpolation is a good algorithm to start with. However, better accuracy and continuous derivatives can be acquired by using a higher order scheme. An often used algorithm is the so-called *bicubic interpolation* [10, § 3.6.3], in which the region between four neighbouring pixels is interpolated. The region is called the *interpolation block*. The four pixels are assumed to be on an unit square $[0,1] \times [0,1]$.

What we are basically looking for is an polynomial of the form

$$f(x,y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j \tag{3.27}$$

in which a_{ij} are the 16 unknowns of the problem. To solve this, we need to have 16 equations. The first four are easily found, since we know the values of f at the corner pixels. The other 12 come from the derivatives f_x, f_y and f_{xy} at those points, in which f_x denotes $\frac{\partial f}{\partial x}$. These 12

equations can be found by differentiating equation 3.27,

$$f_x(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} i a_{ij} x^{i-1} y^j = \sum_{i=1}^{3}\sum_{j=0}^{3} i a_{ij} x^{i-1} y^j \tag{3.28a}$$

$$f_y(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} j a_{ij} x^i y^{j-1} = \sum_{i=0}^{3}\sum_{j=1}^{3} j a_{ij} x^i y^{j-1} \tag{3.28b}$$

$$f_{xy}(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} i j a_{ij} x^{i-1} y^{j-1} = \sum_{i=1}^{3}\sum_{j=1}^{3} i j a_{ij} x^{i-1} y^{j-1} \tag{3.28c}$$

and equate them to the derivatives at the corner points. The total of 16 equations, 3.27 and 3.28, can be written in a system of linear equations. To solve these equations, define α and β to be

$$\alpha = \begin{pmatrix} a_{00} & a_{10} & a_{20} & a_{30} & a_{01} & a_{11} & a_{21} & a_{31} & a_{02} & a_{12} & a_{22} & a_{32} & a_{03} & a_{13} & a_{23} & a_{33} \end{pmatrix}^T \tag{3.29}$$

$$\beta = \begin{pmatrix} f(0,0) & f(1,0) & f(0,1) & f(1,1) & f_x(0,0) & f_x(1,0) & f_x(0,1) & f_x(1,1) & \cdots \\ \cdots & f_y(0,0) & f_y(1,0) & f_y(0,1) & f_y(1,1) & f_{xy}(0,0) & f_{xy}(1,0) & f_{xy}(0,1) & f_{xy}(1,1) \end{pmatrix}^T \tag{3.30}$$

and let $\mathbf{A}$ be a 16×16 matrix, of which the elements can be determined from the 16 equations. This gives the equation $\mathbf{A}\alpha = \beta$. Solving for α, and inserting the values for $\mathbf{A}$, we have to solve equation 3.32 to obtain the interpolation coefficients for this interpolation block.

Note that the bicubic interpolation algorithm does not provide the derivative values at the corner points. Another method has to be used to find them. An approximation can be made by considering the surrounding pixels and is called the *derivative approximation by finite differences* [11]. This gives the following formula's for the derivatives:

$$f_x(x_i,y_j) = \frac{1}{2}\left[f(x_{i+1},y_j) - f(x_{i-1},y_j)\right] \tag{3.31a}$$

$$f_y(x_i,y_j) = \frac{1}{2}\left[f(x_i,y_{j+1}) - f(x_i,y_{j-1})\right] \tag{3.31b}$$

$$f_{xy}(x_i,y_j) = \frac{1}{4}\left[f(x_{i+1},y_{j+1}) - f(x_{i+1},y_{j-1}) - f(x_{i-1},y_{j+1}) + f(x_{i-1},y_{j-1})\right] \tag{3.31c}$$

This means that we need the values of a 4×4 block of pixels to determine the derivatives of a 2×2 block of pixels. This is shown in figure 3.7.

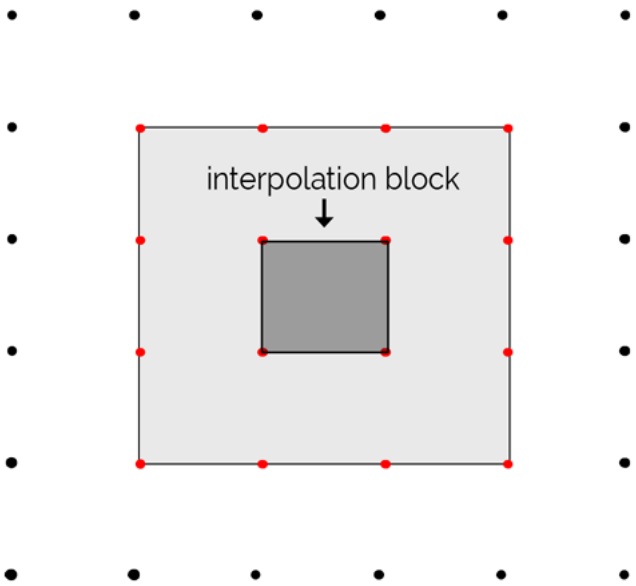


Figure 3.7: A schematic view of an interpolation block. We need all 16 red pixels, a 4x4 block, for the interpolation. These are highlighted with a light grey background. The 2x2 block of pixels, with a dark grey background, is the interpolation block.

$$\alpha = \mathbf{A}^{-1}\beta =$$

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-3 & 3 & 0 & 0 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & -2 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 3 & 0 & 0 & -2 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 & 1 & 1 & 0 & 0 \\
-3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -1 & 0 \\
9 & -9 & -9 & 9 & 6 & 3 & -6 & -3 & 6 & -6 & 3 & -3 & 4 & 2 & 2 & 1 \\
-6 & 6 & 6 & -6 & -3 & -3 & 3 & 3 & -4 & 4 & -2 & 2 & -2 & -2 & -1 & -1 \\
2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
-6 & 6 & 6 & -6 & -4 & -2 & 4 & 2 & -3 & 3 & -3 & 3 & -2 & -1 & -2 & -1 \\
4 & -4 & -4 & 4 & 2 & 2 & -2 & -2 & 2 & -2 & 2 & -2 & 1 & 1 & 1 & 1
\end{pmatrix}
\begin{pmatrix}
f(0,0) \\ f(1,0) \\ f(0,1) \\ f(1,1) \\ f_x(0,0) \\ f_x(1,0) \\ f_x(0,1) \\ f_x(1,1) \\ f_y(0,0) \\ f_y(1,0) \\ f_y(0,1) \\ f_y(1,1) \\ f_{xy}(0,0) \\ f_{xy}(1,0) \\ f_{xy}(0,1) \\ f_{xy}(1,1)
\end{pmatrix} \tag{3.32}$$

Solving equation 3.32 gives an interpolation for one interpolation block. To find an interpolation for the complete image, the complete process has to be done for every interpolation block in the image. At the edges, we have to take into account that finding the derivatives is tricky, since we don't have a 4×4 block to approximate the derivatives. That can be solved by either approximating the values of pixels outside the image or by skipping the boundary interpolation blocks, making the analyzed area smaller but the results more accurate.

Chapter 4

Implementation in Matlab

In the first section, the decision for using home-made solution in Matlab instead of an open-source solution is discussed. The second section elaborates on how the program works internally.

4.1 Implementation decision

Home made versus open source

There exist open-source packages for DIC, using IC-GN as well, for Matlab and other programming languages. The most important reason why the decision fell on a home made solution is that, when programming yourself, every assumption has to be taken implicitly. The result of this is that we know all in and outs of our implementation. When using someone else's code, you can read in the code what he/she decided on, but it is easy to miss essential details. Secondly, a home-made solution would never bring troubles with the possibility of commercializing the 3D SEM-DIC in the back of our minds. Third, programming it yourself would be a better learning experience and is, not completely unimporant, way more fun.

Matlab versus C/C++

The next decision was in which programming language the algorithms would be implemented. It boiled down to either Matlab or C/C++ pretty fast. The pro's for Matlab are it's ease of use and wide availability of numerical tools, making development time a magnitude shorter. The advantage of C or C++ is their computational efficiency, making the computation probably a magnitude shorter. However, the development is more complex and time consuming, due to

the fact that there are less built in tools and you have to take care of memory management.

For this project, Matlab was chosen due to the shorter development time and easy prototyping. When developing a production ready DIC program, prototyping in Matlab and porting the algorithms to C/C++ for speed might be the best solution. As it turns out, running the IC-GN algorithm in Matlab takes several hours and is therefore less useful for processing a lot of images. Expectations are that an implementation in C or C++ would run in less than 5 minutes, but this should be investigated further.

4.2 Program flow

The program is written in the imperative paradigm and consists of several sections, which are all elaborated in the paragraphs below. The main program flow is shown in figure 4.1

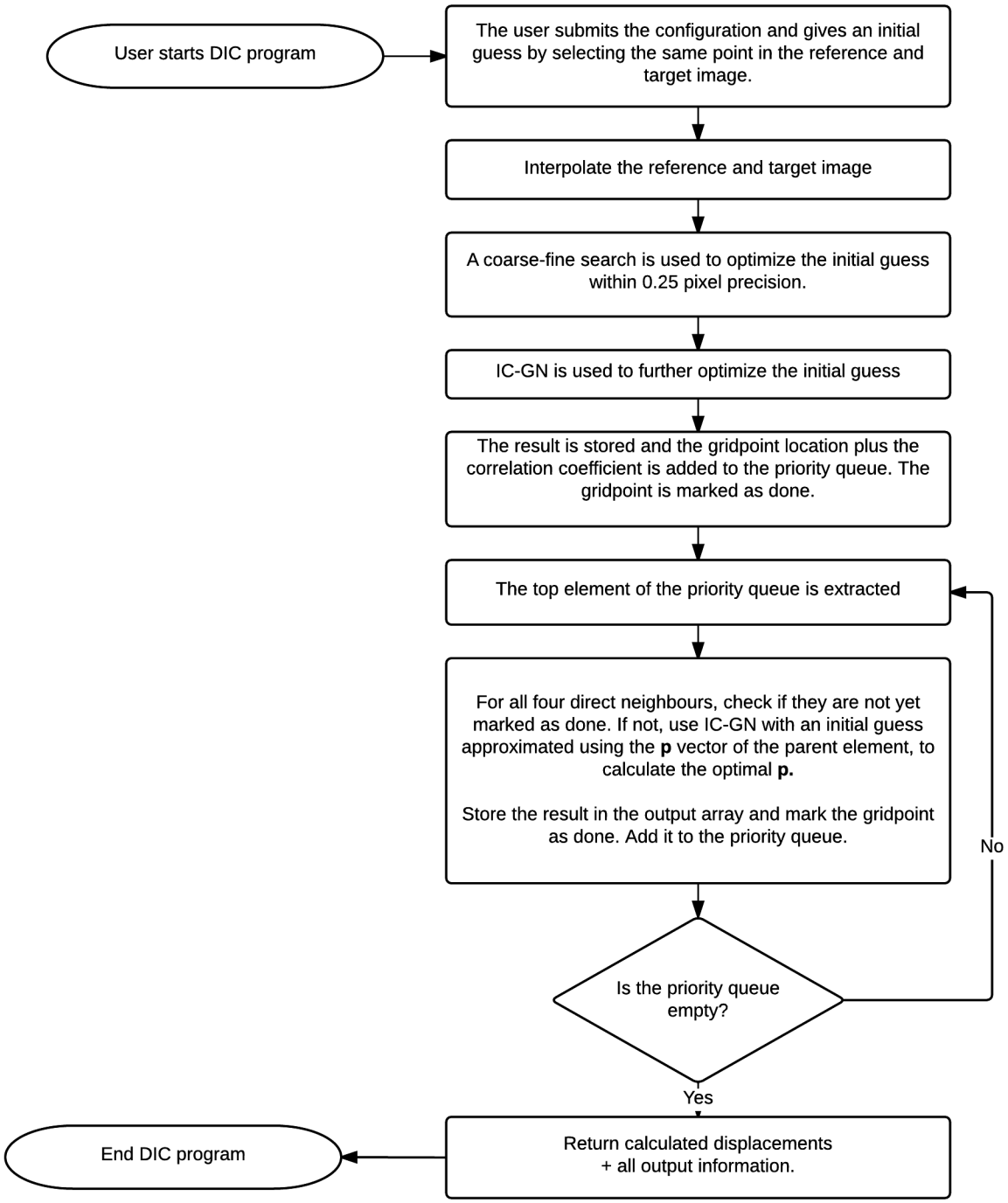


Figure 4.1: A global flowchart of the program

Configuration and the initial guess

After the user starts the program, configuration need to e supplied. All available options are explained in table 4.1. After this, the user needs to supply an initial guess to help set up the algorithm. The user is a great tool for this, since it has the availaility with the zoom function of Matlab images to select two points in different image within one pixel precision. The process works by showing the reference image with the center gridpoint highlighted, and letting the user select the same point / pixel in the target image.

Option	**Explanation**
Reference image location	The file location of the reference image.
Target image location	The file location of the target image.
Subset size	Both the width and height of the subset. This choice influences the number of gridpoints analyzed, since a larger subset results in a larger margin betweem the outer gridpoints and the image border. A larger subset size results in longer computation time, but the correlation is based on a larger area.
Gridspacing	Both the number of rows and columns for the grid spacing. The number of gridpoints, naturally, also depend on this. More gridpoints mean a larger computing time, but also more information.
Convergence criterion	When $\|\delta\mathbf{p}\|$ is smaller than the convergence criterion, IC-GN stops optimizing and returns. A smaller convergence criterion might result in more accurate results.
Max. number of iterations	After this number of iterations is reached in IC-GN, the optimization stops and the result is *not convergent*. This is used to prevent infinite looping for not convergent solutions.
Pixel precision	This setting determines the step we pixel step we take in the correlation algorithm. A smaller pixel precision increases computing time, but also the accuracy of the match.

Table 4.1: All available options for the program

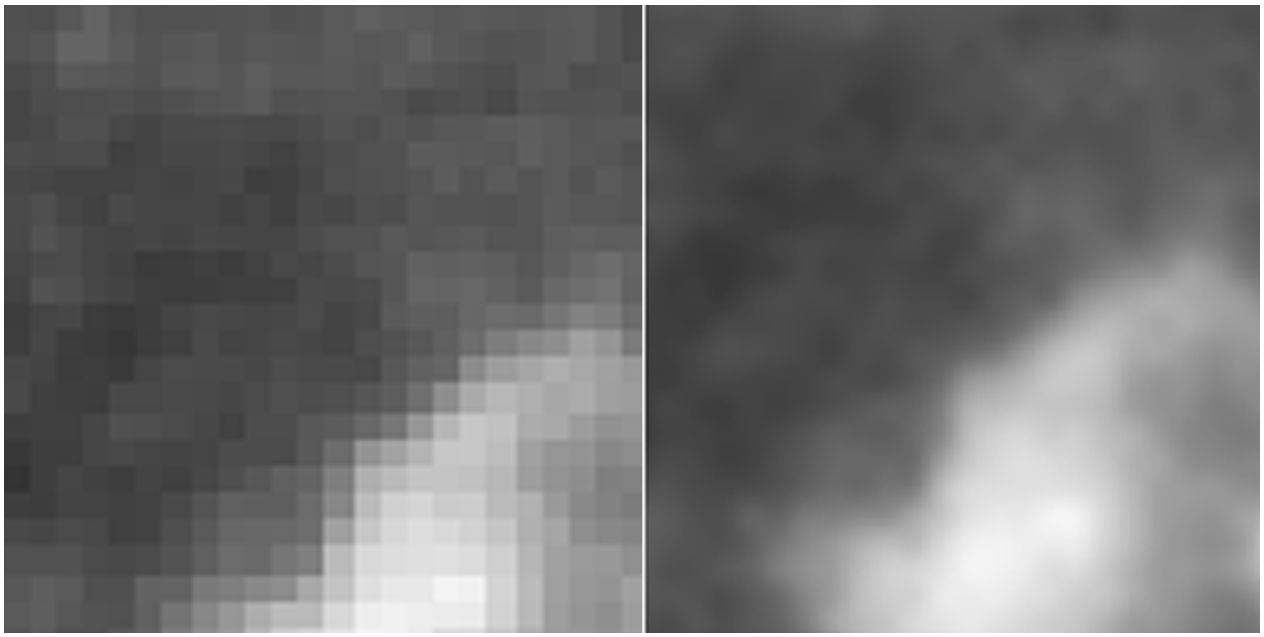


Figure 4.2: The result of interpolation. The left image shows the original image, the right image is after interpolation and plotted on a grid 64 times as dense. It can unfortunately not be shown that we have a continous image, due to the discrete nature of printed images.

Interpolation

The next step is to interpolate both the reference and target image. The complete images are interpolated and the coefficients are stored in a way that they can always be accessed by other parts of the program, so the interpolation only has to be done once. The results of interpolation are shown in figure 4.2.

Our solution is verified by comparing the results to Matlab's bicubic convolution implementation. The way this was done was by plotting the same row of greyvalues for our algorithm and Matlab's. The results are shown in figure 4.3. It can be seen that the results are exactly equiavlent. One could wonder why we bothered to implement our own interpolation algorithm. The reason for this originates from the project requirements - we know exactly how and what we implemented.

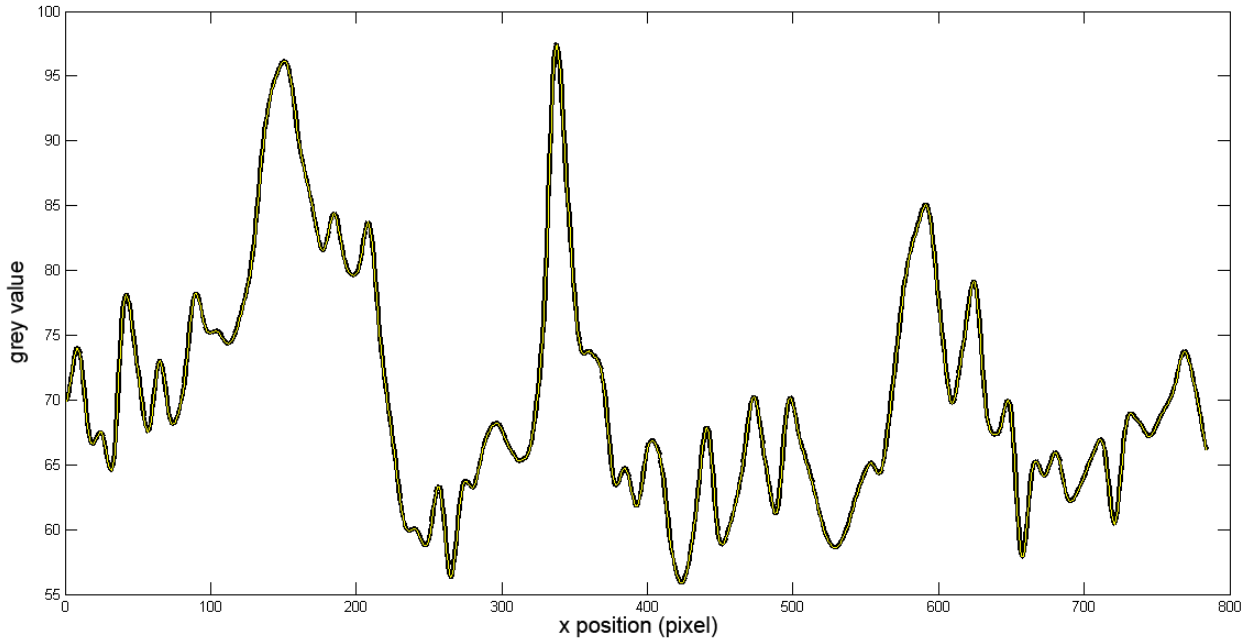


Figure 4.3: A comparison between Matlab's cubic interpolation function *interp2* and our implementation. The same pixel row in both results are plotted. The two lines are completely overlapping. If both lines are plotted with the same thickness, they cannot be distinguished.

Optimizing initial guess using coarse fine search

The result of the IC-GN algorithm heavily depends on the accuracy of the initial guess. The guess provided by the user is already quite good, but can and should be be improved. Therefore, a coarse-fine search in a region of 3x3 pixels around the initial guess is done, with a default pixel precision of 0.25. This step is still relatively large, and the is only done in the u and v parameters, not their derivatives. The reason for this is that the coarse fine search is extremely time consuming. However, it should be investigated whether a more extensive coarse-fine search results in better correlation results.

IC-GN using a reliability guided displacement tracking strategy

Next, the result of the coarse fine search is optimized using IC-GN. For this initial gridpoint, the final **p** vector, correlation coefficient, number of required iterations are stored in an output array. An indicator for the result is also stored, i.e. a convergent result, not convergent or that the displacement vector points to a location out of the target image. The program flow of the IC-GN algorithm is shown in figure 4.4.

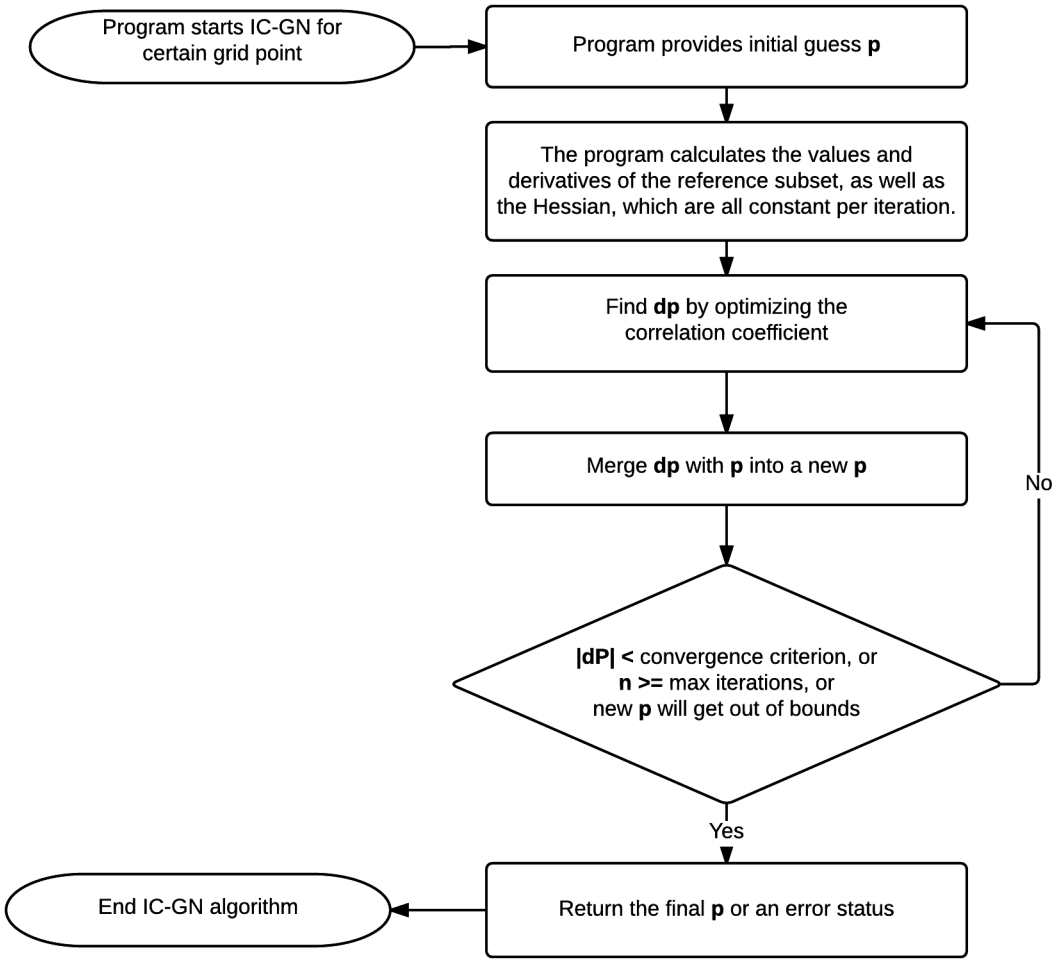


Figure 4.4: A global flowchart of the IC GN algorithm. In this diagram, **dp** equals δ**p** and **n** the number of iterations done.

The question at this point is how we are going to provide an initial guess for the other gridpoints. Repeating the above process would be inefficient. Therefore, a reliability guided displacement tracking strategy is employed [8]. The first step is to store the results of the initial gridpoint in a priority queue, a data structure which orders the elements inside of it on some property. The property that this priority queue sorts on is the correlation coefficient, where the lowest coefficients are stored in the front of the queue.

While the queue is not empty, the algorithm pops the first gridpoint of the queue. That will be the gridpoint with the lowest coefficient we have encountered so far. The resulting **p** vector of the that gridpoint is used as the ingredient for the initial guess for all four direct neighbours i.e. north, south, east and west. We then analyze these gridpoints, if they haven't already been,

with an initial guess aquired by using the following transformation:

$$u_{guess} = u + u_x \Delta x + u_y \Delta y \tag{4.1}$$

$$v_{guess} = v + v_x \Delta x + v_y \Delta y \tag{4.2}$$

$$u_{x,guess} = u_x \tag{4.3}$$

$$u_{y,guess} = u_y \tag{4.4}$$

$$v_{x,guess} = v_x \tag{4.5}$$

$$v_{y,guess} = v_y \tag{4.6}$$

where Δx and Δy are either zero or defined by the gridsize (depending on whether you analyze the west/east or north/south neighbour). The results are stored in the queue, thereby providing new seed points for the algorithm. If you continue this process, by the time the queue is empty, all gridpoints have been analyzed. Note that this algorithm is a so-called *greedy algorithm*, which are defined as *algorithms that follow the problem solving heuristic of making the locally optimal choice at each stage*. We won't know for sure that another neighbour of a gridpoint, with a slightly higher coefficient, would not result in a better displacement vector. However, it is probably in most cases our best choice, so we use it nevertheless.

Chapter 5

Discussion

During the development of the DIC algorithm, we encountered several discussion points, in the form of theorethical assumptions and possible improvements for the algorithm. They are discussed in the section below and could be analyzed in further studies.

5.1 Theoretical

First order approximations

In several mathematical derivations we used first order approximations. First of all, the displacement vector **p** is only in the first order. In a small subset the second order would probably have negligible influence, while a larger subset provides better correlation. However, is the second order derivative in displacements in larger subsets also negligible?

Secondly, the derivation of the IC-GN uses a Taylor expansion in which the second order terms and higher are dropped. This results in an optimization step which is highly efficient in terms of computing time, but some information is dropped. Even so, it is expected that the computational advantage is lost when the second order is also taken into account, since the Hessian will then depend on **p** again. It should be investigated what the implications of dropping the higher terms are and what the improvement, or extra information. of the algorithm is when they are taken back into account.

Error quantification

In the current algorithm, no error indication is given in any quantified form. This is a requirement of the project, but is not yet implemented. There are a few characteristics that could pro-

vide information about the error. First of all, the final correlation coefficient. The ZNSSD coefficent is in the range [0, 4], in which 0 is the best. A coefficient as close as possible to 0 could mean a smaller error. Secondly, the "width" of the minimum (e.g. half-width minimum) in 6 dimensions could provide information about how sure we are that the found displacement is the correct one. If the minimum is very wide, chances are the real displacement could be a bit different. Information extracted from the algorithm, i.e. the number of iterations or the change in $\delta\mathbf{p}$, i.e. $|\delta\mathbf{p}|$, before breaking the loop could also provide error indications.

Intensity gradients

Currently, by the *zero mean* and *normalization* additions to the correlation coefficient, brightness offset and scale changes are taken into account. However, the scanning nature of SEM allows for gradients in the brightness as well. An improved algorithm would take this gradient into account.

Integrals instead of discrete steps for correlation

After interpolation, a continuous image is available. This means that we won't necessarily need to do discrete steps in the correlation procedure - we can use integrals instead. This would have significant impact on the computing time and have better results. We explored this idea briefly, but came across one thing pretty quickly: the integrals are extremely long and difficult, having to take the 6 dimensions of **p**into account. However, the ideas are still promising and it would be break-through if this worked out.

5.2 Implementation

Minimization algorithm

As to quote numerical recipes [10],

> In the next chapter, we will find that there are efficient general techniques for finding a minimum of a function of many variables. Why is that task (relatively) easy, while multidimensional root finding is often quite hard? Isn't minimization equivalent to finding a zero of an N-dimensional gradient vector, which is not so different from zeroing an N-dimensional function? No! The components of a gradient vector are not independent, arbitrary functions.

Currently we are using multidimensional root finding, and get accurate results since we have a good initial guess. however, apparently there exist algorithms which can do a better job in finding the minimum in our six dimensions. This should be investigated as to improve our results.

Improve the initial guess

Currently, the initial guess by the user is improved using a coarse-fine search, after which it is fed into the IC-GN algorithm. It is known that the IC-GN gives better results when the initial guess is better. However, currently the implementation only optimizes u and v in steps of 0.25 pixels on a 3×3 block of pixels. This is partly due to development considerations, since coarse-fine search is really slow and optimizing a lot makes testing awful, but we also don't have a real idea of how far we should go with our initial guess, and where the ideal tradeoff between computing time and accuracy lies. It should be investigated whether optimizing the derivatives of u and v, as well as optimizing in smaller steps, has significant influence on the final result.

Higher order interpolation scheme

Currently we are using an third-order polynomial for our interpolation scheme. Higher order polynomials might be better suited to model the edges of bright and dark spots in the SEM images, which might result in better results since correlation can be done more accurately. There probably exist fourth or fifth order interpolation solutions, so a literature study in that field might be beneficial.

Handling not convergent images

Currently, it is possible that the algorithm gets "stuck" when it needs to traverse a region without many points of recognition, i.e. a dark grey area. Then all gridpoints will not converge since equally good matches are found with different $\delta\mathbf{p}$, since two grey regions are more or less the same. This can be solved by (sprayed) speckle patterns on the sample, but that is probably suboptimal. This challenge could probably also be solved in different, smarter ways, when some thought is put in it. One could for example "jump over" difficult regions and re-apply coarse fine search to get a good initial guess, using the displacement from the gridpoint before the jump as an initial guess.

Chapter 6

Conclusions

After a literature study in DIC, the IC-GN algorithm was implemented, employing a reliability-guided displacement tracking strategy. The ZNSSD correlation criterion was used, which takes brightness offset and scale changes into account. Bicubic interpolation is used to provide sub-pixel displacements. The implementation was done in Matlab due to its wide range of numerical tools availale and easy prototyping.

Several improvements can be made to the algorithm, since the final result of this project is (just) an early prototype. Error quantification has to be built in and the algorithm needs to be tested with actual SEM images. Furthermore, several considerations and assumptions needs to be analyzed and discussed as to improve the robustness of the algorithm. To make the software production ready, it probably needs to be proted to C or C++ or any other low level language, due to the computational inefficiency of Matlab.

Appendix A

Appendices

A.1 Deriving the optimization step for $\delta\mathbf{p}$ in the IC-GN algorithm

We want to minimze the following equation with respect to $\delta\mathbf{p}$,

$$C_{ZNSSD} = \sum \left(\frac{f(\mathbf{x}+\mathbf{W}(\mathbf{x},\delta\mathbf{p})) - \bar{f}}{\sqrt{\sum \bar{f}_{ij}^2}} - \frac{g(\mathbf{x}+\mathbf{W}(\mathbf{x},\mathbf{p})) - \bar{g}}{\sqrt{\sum \bar{g}_{ij}^2}} \right)^2 \tag{A.1}$$

To do this, we first perform a first order Taylor expansion with respect to $\delta\mathbf{p}$, which results, together with some algebra, in

$$C_{ZNSSD} = \sum \left(\left(f(\mathbf{x}) - \nabla f \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \delta\mathbf{p} - \bar{f} \right) - \frac{\Delta f}{\Delta g} g(\mathbf{x}+\mathbf{W}(\mathbf{x},\mathbf{p})) - \bar{g} \right)^2 \tag{A.2}$$

Minimizing equation A.2 with respect to $\delta\mathbf{p}$, i.e. $\frac{\partial C_{ZNSSD}}{\partial(\delta\mathbf{p})} = \mathbf{0}$, is a least-squares problem. In general, for a least-square equation of the form

$$S = \sum_i r^2 \tag{A.3}$$

the solution is, assuming $\mathbf{r}$ is a function of β, of the form

$$\frac{\partial S}{\partial \beta_j} = 2 \sum_i r \frac{\partial r}{\partial \beta_j} \tag{A.4}$$

Getting back to our case, we have

$$r = \left(f(\mathbf{x}) - \nabla f \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \delta \mathbf{p} - \bar{f}\right) - \frac{\Delta f}{\Delta g} g(\mathbf{x} + \mathbf{W}(\mathbf{x}, \mathbf{p})) - \bar{g} \tag{A.5}$$

$$\frac{\partial r}{\partial \beta_j} \equiv \frac{\partial r}{\partial (\delta \mathbf{p})} = \left(\nabla f \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right)^T \tag{A.6}$$

Plugging everything back into A.4, we get

$$\frac{\partial C_{ZNSSD}}{\partial (\delta \mathbf{p})} = 2 \sum \left(\nabla f \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right)^T \left\{\left(f(\mathbf{x}) - \nabla f \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \delta \mathbf{p} - \bar{f}\right) - \frac{\Delta f}{\Delta g} g(\mathbf{x} + \mathbf{W}(\mathbf{x}, \mathbf{p})) - \bar{g}\right\} \tag{A.7}$$

$$= 2 \sum \left(\nabla f \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right)^T \left(\nabla f \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right) \delta \mathbf{p} + \left(\nabla f \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right)^T \left\{f(\mathbf{x}) - \bar{f} - \frac{\Delta f}{\Delta g} g(\mathbf{x} + \mathbf{W}(\mathbf{x}, \mathbf{p})) - \bar{g}\right\} \tag{A.8}$$

$$= 2 \sum \mathbf{H} \delta \mathbf{p} + \left(\nabla f \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right)^T \left\{f(\mathbf{x}) - \bar{f} - \frac{\Delta f}{\Delta g} g(\mathbf{x} + \mathbf{W}(\mathbf{x}, \mathbf{p})) - \bar{g}\right\} \tag{A.9}$$

$$= \mathbf{0} \tag{A.10}$$

with $\mathbf{H} = \left(\nabla f \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right)^T \left(\nabla f \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right)$. Solving this equation for $\delta \mathbf{p}$, we get our final result

$$\delta \mathbf{p} = -\sum \mathbf{H}^{-1} \times \sum \left\{\left(\nabla f \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right)^T \left(f(\mathbf{x}) - \bar{f} - \frac{\Delta f}{\Delta g} g(\mathbf{x} + \mathbf{W}(\mathbf{x}, \mathbf{p})) - \bar{g}\right)\right\} \tag{A.11}$$

Bibliography

[1] E.T. Faber, D. Martinez-Martinez, C. Mansilla, V. Ocelìk, and J.Th.M. De Hosson. Calibration-free quantative surface topography reconstruction in scanning electron microscopy. *Ultramicroscopy*, 148:31–41, 2015.

[2] Jeol. Invitation to the sem world.

[3] H Exner. Scanning electron microscopy.

[4] T.C. Chu, W.F. Ranson, and M.A. Sutton. Applications of digital-image-correlation techniques to experimental mechanics. *Experimental Mechanics*, 25(3):232–244, 1985.

[5] H.A. Bruck, S.R. McNeill, M.A. Sutton, and III Peters, W.H. Digital image correlation using newton-raphson method of partial differential correction. *Experimental Mechanics*, 29(3):261–267, 1989.

[6] G. Vendroux and W.G. Knauss. Submicron deformation field measurements: Part 2. improved digital image correlation. *Experimental Mechanics*, 38(2):86–92, 1998.

[7] Bing Pan, Huimin Xie, and Zhaoyang Wang. Equivalence of digital image correlation criteria for pattern matching. *Appl. Opt.*, 49(28):5501–5509, Oct 2010.

[8] B. Pan, K. Li, and W. Tong. Fast, robust and accurate digital image correlation calculation without redundant computations. *Experimental Mechanics*, 53(7):1277–1289, 2013.

[9] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. 1:I–1090–I–1097 vol.1, 2001.

[10] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes - The Art of Scientific Computing*. Cambridge University Press, third edition, 2007.

[11] C. Eberly. Derivative approximation by finite differences. http://math.nyu.edu/ atm262/fall06/compmethods/a1/DerivativesApproximationByFiniteDifferences.p 2003.