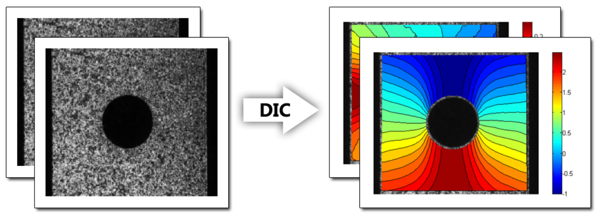
DIC Algorithms

Table of Contents

1 - DIC Analysis

The intent of this write up is to provide an explanation of the mathematical framework and modern algorithms used in Ncorr, and in 2D-DIC in general. If the reader is already familiar with DIC and would simply like to use Ncorr or experiment with it first, it is available for download in the downloads section of this website.

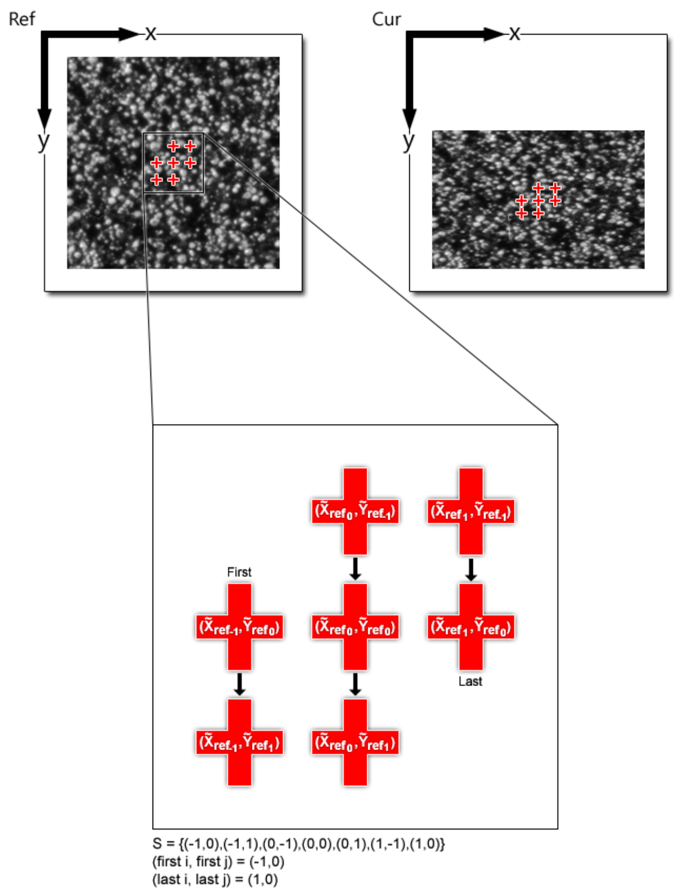
1.1 - The general problem



The overall goal of DIC is to obtain displacement and strain fields within a region of interest (ROI) for a material sample undergoing deformation. DIC uses image processing techniques in an attempt to solve this problem. Basically, images of a sample are taken as it deforms; these images are used as inputs to a DIC program. The idea is to somehow obtain a one-to-one correspondence between material points in the reference (initial undeformed picture) and current (subsequent deformed pictures) configurations. DIC does this by taking small subsections of the reference image, called subsets, and determining their respective locations in the current configuration. For each subset, we obtain displacement and strain information through the transformation used to match the location of the subset in the current configuration. Many subsets are picked in the reference configuration, often with a spacing parameter to reduce computational cost (also note that subsets typically overlap as well). The end result is a grid containing displacement and strain information with respect to the reference configuration, also referred to as Lagrangian displacements/strains. The displacement/strain fields can then either be reduced or interpolated to form a "continuous" displacement/strain field. These ideas will be made more precise in the following sections.

2 - Computational Implementation

To be more specific, subsets are essentially a group of coordinate points; the idea of subsets in the reference and current image is shown in fig.1.


Ref
X
y
Cur
X
y
First
Last
$(\tilde{X}_{ref_0}, \tilde{Y}_{ref_{-1}})$
$(\tilde{X}_{ref_1}, \tilde{Y}_{ref_{-1}})$
$(\tilde{X}_{ref_{-1}}, \tilde{Y}_{ref_0})$
$(\tilde{X}_{ref_0}, \tilde{Y}_{ref_0})$
$(\tilde{X}_{ref_1}, \tilde{Y}_{ref_0})$
$(\tilde{X}_{ref_{-1}}, \tilde{Y}_{ref_1})$
$(\tilde{X}_{ref_0}, \tilde{Y}_{ref_1})$

S = {(-1,0),(-1,1),(0,-1),(0,0),(0,1),(1,-1),(1,0)}
(first i, first j) = (-1,0)
(last i, last j) = (1,0)

Fig.1. The subset's coordinates are shown as red crosses. The notation at the bottom is used throughout the rest of this document. The subset does not need to be square, as shown above; the shape can be arbitrary as long as it contains a center point.

The transformation of initial reference subset points to the current configuration is typically constrained to a linear, first order transformation (although second order transformations can also be used if very large strains are anticipated[1]) as shown below:

$$\tilde{x}_{cur_i} = x_{ref_i} + u_{rc} + \frac{\partial u}{\partial x_{rc}}\left(x_{ref_i} - x_{ref_c}\right) + \frac{\partial u}{\partial y_{rc}}\left(y_{ref_j} - y_{ref_c}\right) \qquad (i,j) \in S \qquad \text{Eq.1}$$

$$\tilde{y}_{cur_j} = y_{ref_j} + v_{rc} + \frac{\partial v}{\partial x_{rc}}\left(x_{ref_i} - x_{ref_c}\right) + \frac{\partial v}{\partial y_{rc}}\left(y_{ref_j} - y_{ref_c}\right)$$

$$\boldsymbol{p} = \left\{u \quad v \quad \frac{\partial u}{\partial x} \quad \frac{\partial u}{\partial y} \quad \frac{\partial v}{\partial x} \quad \frac{\partial v}{\partial y}\right\}^T$$

Eq.2

Where x_{ref_i} and y_{ref_j} are the x and y coordinates of an initial reference subset point, x_{ref_c} and y_{ref_c} are the x and y coordinates of the center of the initial reference subset, $\tilde{x}_{cur_i}$ and $\tilde{y}_{cur_j}$ are the x and y coordinates of a final current subset point, (i,j) are indices used for the relative location of the subset points with respect to the center of the subset, as well as for correspondences between subset points in the current and reference configuration, and S is a set which contains all of the subset points (an example is given in fig.1). The subscript "rc" used in eq.1 is meant to signify that the transformation is from the reference to the current coordinate system. Furthermore, $\{u, v, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y}\}^T$ is the general form of the deformation vector, **p**, as defined in eq.2 (its effect on subsets is shown in fig.2). Lastly, eq.1 can also be written in matrix form as shown below:

$$\xi_{\mathrm{ref_c}} + w\left(\Delta\xi_{ref}, \mathrm{p_{rc}}\right) = \begin{Bmatrix} x^{\mathrm{T}}_{\mathrm{ref_c}} \\ y^{\mathrm{T}}_{\mathrm{ref_c}} \\ \mathbf{1} \end{Bmatrix} + \begin{bmatrix} 1 + \frac{\mathrm{du}}{\mathrm{dx_{rc}}} & \frac{\mathrm{du}}{\mathrm{dy_{rc}}} & \mathrm{u_{rc}} \\ \frac{\mathrm{dv}}{\mathrm{dx_{rc}}} & 1 + \frac{\mathrm{dv}}{\mathrm{dy_{rc}}} & \mathrm{v_{rc}} \\ 0 & 0 & 1 \end{bmatrix} * \begin{Bmatrix} \Delta x^{\mathrm{T}}_{\mathrm{ref}} \\ \Delta y^{\mathrm{T}}_{\mathrm{ref}} \\ \mathbf{1} \end{Bmatrix}$$

Eq.3

Where ξ is an augmented vector containing the x and y coordinates of subset points, Δx and Δy are the distance between a subset point and the center of the subset,and "w" is a function called a warp. For computational purposes, we also allow the reference subset to deform within the reference configuration, as shown below:

$$\tilde{x}_{ref_i} = x_{ref_i} + u_{rr} + \frac{\partial u}{\partial x_{rr}}\left(x_{ref_i} - x_{ref_c}\right) + \frac{\partial u}{\partial y_{rr}}\left(y_{ref_j} - y_{ref_c}\right)$$

$$\tilde{y}_{ref_j} = y_{ref_j} + v_{rr} + \frac{\partial v}{\partial x_{rr}}\left(x_{ref_i} - x_{ref_c}\right) + \frac{\partial v}{\partial y_{rr}}\left(y_{ref_j} - y_{ref_c}\right)$$

$(i, j) \in S$

Eq.4

Where $\tilde{x}_{ref_i}$ and $\tilde{y}_{ref_j}$ are x and y coordinates of a final reference subset point. The "rr" subscript in eq.4 is meant to signify that the transformation is from the reference coordinate system to the reference coordinate system.

For our purposes, *we want to find the optimal* $\mathbf{p}_{rc}$*, when* $\mathbf{p}_{rr} = \mathbf{0}$*, such that the coordinates at* $\tilde{x}_{ref_i}$ *and* $\tilde{y}_{ref_j}$ *best match the coordinates at* $\tilde{x}_{cur_i}$ *and* $\tilde{y}_{cur_j}$*, respectively*. I denote this $\mathbf{p}_{rc}$ value $\mathbf{p}^{*}_{rc}$. For now, it may seem nonsensical to even use eq.4 and just restrict the reference subset coordinates such that they are undeformable. But, one of the methods we will eventually use to find $\mathbf{p}^{*}_{rc}$ (the inverse compositional method) requires the reference subset coordinates to be deformable, so I introduced this notation now to make the subsequent derivation smoother.

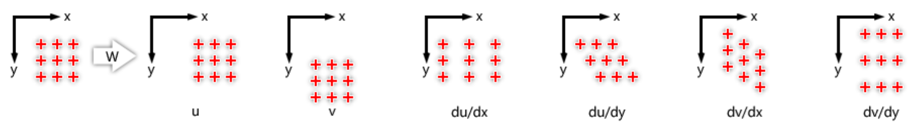


Fig.2. The above figure shows linear transformations for subset coordinates. Any linear combination of the 6 parameters shown above can be used to deform subset coordinates through a warp function, w.

2.1 - Correlation criteria

The next step is to establish a metric for similarity between the final reference subset and final current subset. This is carried out by comparing grayscale values at the final reference subset points with gray scale values at the final current subset points. The following two metrics are the most commonly used in DIC:

$$C_{CC} = \frac{\sum_{(i,j)\in S}\left(f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) - f_m\right)\left(g\left(\tilde{x}_{cur_i}, \tilde{y}_{cur_j}\right) - g_m\right)}{\sqrt{\sum_{(i,j)\in S}\left[f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) - f_m\right]^2 \sum_{(i,j)\in S}\left[g\left(\tilde{x}_{cur_i}, \tilde{y}_{cur_j}\right) - g_m\right]^2}}$$

Eq.5

$$C_{LS} = \sum_{(i,j)\in S}\left[\frac{f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) - f_m}{\sqrt{\sum_{(i,j)\in S}\left[f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) - f_m\right]^2}} - \frac{g\left(\tilde{x}_{cur_i}, \tilde{y}_{cur_j}\right) - g_m}{\sqrt{\sum_{(i,j)\in S}\left[g\left(\tilde{x}_{cur_i}, \tilde{y}_{cur_j}\right) - g_m\right]^2}}\right]^2$$

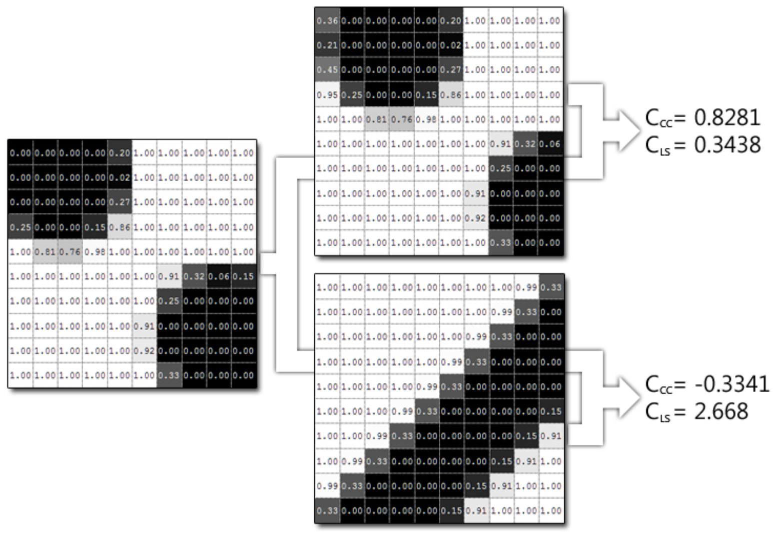
Eq.6

Where f and g are the reference and current image functions, respectively, and return a grayscale value corresponding to the specified (x,y) point. f_m and g_m are the mean grayscale values of the final reference and current subset, respectively, as defined by:

$f_m = \frac{\sum_{(i,j)\in S} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right)}{n(S)}$	Eq.7
$g_m = \frac{\sum_{(i,j)\in S} g\left(\tilde{x}_{cur_i}, \tilde{y}_{cur_j}\right)}{n(S)}$	Eq.8

Where n(S) is the number of elements in S.

Eq.5 is the normalized cross correlation criterion and indicates a good match when C_{CC} is close to 1. Eq.6 is the normalized least squares criterion and indicates a good match when C_{LS} is close to 0. Subtraction of the mean component (f_m and g_m) in eq.5-6 allow for the criteria to be invariant to shifts in grayscale values, while division by the quantity in the denominator allow for invariance with respect to scaled grayscale values; the end result is correlation criteria which are invariant to affine shifts in grayscale values. In actuality, these correlation criteria are directly related [2], but each respective correlation criterion is easier to calculate in certain situations which is why both are listed. A simple example of how both are used is given in fig.3.



0.00	0.00	0.00	0.00	0.20	1.00	1.00	1.00	1.00	1.00
0.00	0.00	0.00	0.00	0.02	1.00	1.00	1.00	1.00	1.00
0.00	0.00	0.00	0.00	0.27	1.00	1.00	1.00	1.00	1.00
0.25	0.00	0.00	0.15	0.86	1.00	1.00	1.00	1.00	1.00
1.00	0.81	0.76	0.98	1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00	0.91	0.32	0.06	0.15
1.00	1.00	1.00	1.00	1.00	1.00	0.25	0.00	0.00	0.00
1.00	1.00	1.00	1.00	1.00	0.91	0.00	0.00	0.00	0.00
1.00	1.00	1.00	1.00	1.00	0.92	0.00	0.00	0.00	0.00
1.00	1.00	1.00	1.00	1.00	1.00	0.33	0.00	0.00	0.00

0.36	0.00	0.00	0.00	0.00	0.20	1.00	1.00	1.00	1.00
0.21	0.00	0.00	0.00	0.00	0.02	1.00	1.00	1.00	1.00
0.45	0.00	0.00	0.00	0.00	0.27	1.00	1.00	1.00	1.00
0.95	0.25	0.00	0.00	0.15	0.86	1.00	1.00	1.00	1.00
1.00	1.00	0.81	0.76	0.98	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.91	0.32	0.06
1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.25	0.00	0.00
1.00	1.00	1.00	1.00	1.00	1.00	0.91	0.00	0.00	0.00
1.00	1.00	1.00	1.00	1.00	1.00	0.92	0.00	0.00	0.00
1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.33	0.00	0.00

1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.33
1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.33	0.00
1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.33	0.00	0.00
1.00	1.00	1.00	1.00	1.00	0.99	0.33	0.00	0.00	0.00
1.00	1.00	1.00	1.00	0.99	0.33	0.00	0.00	0.00	0.00
1.00	1.00	1.00	0.99	0.33	0.00	0.00	0.00	0.00	0.15
1.00	1.00	0.99	0.33	0.00	0.00	0.00	0.00	0.15	0.91
1.00	0.99	0.33	0.00	0.00	0.00	0.00	0.15	0.91	1.00
0.99	0.33	0.00	0.00	0.00	0.00	0.15	0.91	1.00	1.00
0.33	0.00	0.00	0.00	0.00	0.15	0.91	1.00	1.00	1.00

Fig.3. The above figure demonstrates the effectiveness of the two correlation criteria. The grayscale values on the left are compared with the two on the right. The correlation criteria show that the top right gray scale values match the set on the left (C_{CC} is close to 1 and C_{LS} is close to 0), which is obvious through visual inspection.

The general steps for the utilization of these correlation criteria are outlined below:

1. Form an initial subset in the reference configuration, typically at integer pixel locations. Apply a warp, with $\mathbf{p}_{rr}$, to the initial reference subset and sample f, the reference gray scale values, at these points and store these values in an array

2. Apply a warp, with $\mathbf{p}_{rc}$, to the initial reference subset to bring it to the current configuration. Sample g, the current gray scale values, at these points and store them in an equivalent sized array as in step 1 (note that the storing of these points is essentially an inverse transformation and allows for a direct correspondence/comparison between grayscale values in the reference and current configuration).

3. Now you can compare these two arrays using either of the two correlation criteria in eqs.5-6. This process is outlined in the figure below:

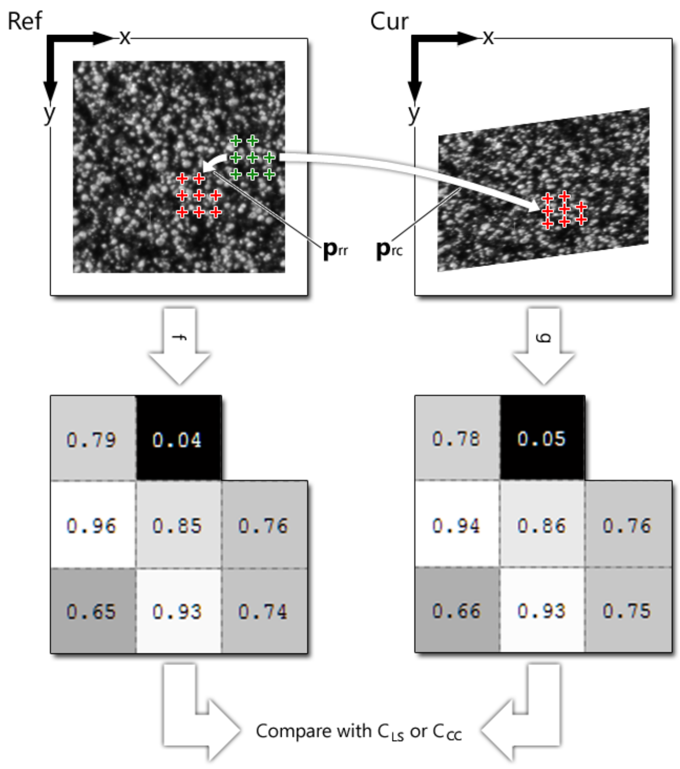


Fig.4. This figure is a graphical example of the steps outlined above. The green subset in the reference image is the initial reference subset. The red subsets mark the final subset locations in the reference and current configuration.

Note that if $\mathbf{p}_{rr}$ and $\mathbf{p}_{rc}$ are found so that the final subsets match well, and $\mathbf{p}_{rr}$ is small in translational magnitude, then $\mathbf{p}_{rc}^{*}$ can be approximated by taking $\mathbf{p}_{rc}$ and composing it with the inverse of $\mathbf{p}_{rr}$. Imagine this as a direct transformation from the final reference subset to the final current subset. This idea is important for an understanding of the inverse compositional algorithm which is discussed later.

Anyway, the next step of the analysis is how to establish a method to find $\mathbf{p}_{rc}^{*}$ in an automated way. The main implementation in DIC is to use an iterative nonlinear least squares optimization scheme which ***minimizes*** eq.6 (note that eq.6 is strictly positive and a smaller number indicates a better match).

3 - Nonlinear Optimization

The nonlinear optimization section is broken into three main parts: 1) providing an initial guess, 2) the Gauss-Newton (GN) iterative optimization scheme and 3) interpolation. An initial guess is required because iterative optimization schemes converge to a *local* maximum/minimum; thus, an initial guess near the *global* maximum/minimum is required. In the GN iterative optimization scheme section, two different schemes are discussed: the forward additive Gauss-Newton method (FA-GN), and the inverse compositional Gauss-Newton method (IC-GN). The FA-GN method is discussed first because this is a commonly utilized algorithm in DIC (and in optimization in general) . On the other hand, the IC-GN method is a special case of nonlinear optimization (commonly utilized in image alignment algorithms) which is faster than the standard FA-GN method and was recently applied to DIC [**3**]. Lastly, interpolation is discussed because grayscale values and gradients at subpixel locations are required to compute quantities used in the FA-GN and IC-GN methods.

The general flow of a nonlinear optimization scheme is shown below:

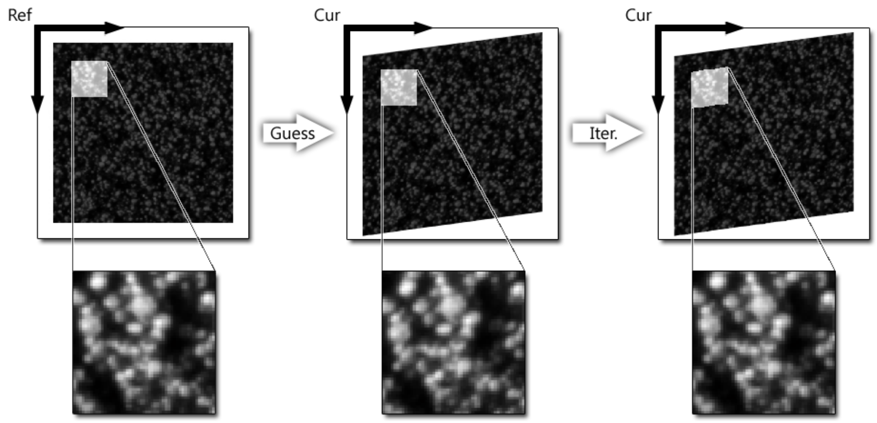


Fig.5. The figure above outlines the basic steps of the nonlinear optimization scheme used in DIC. The first step is to obtain the initial guess. This guess is fed as the initial input to the iterative optimization scheme, which provides more accurate results, as shown in the last picture on the right.

3.1 - The initial guess

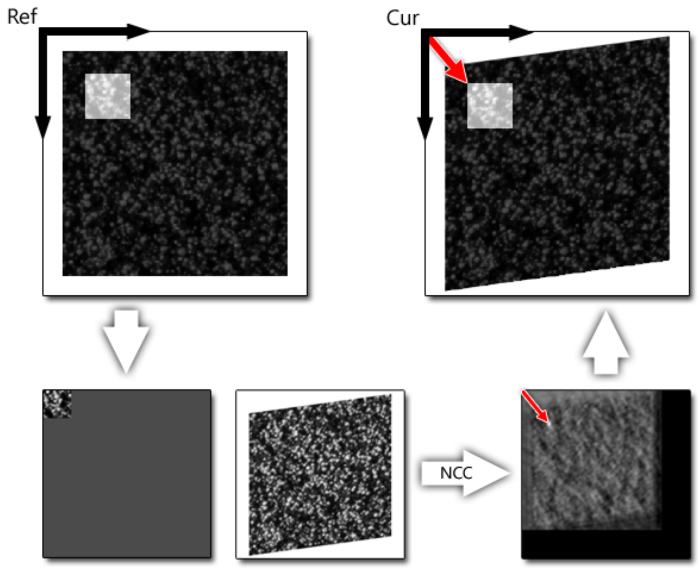


Fig.6. The above figure demonstrates normalized cross correlation (eq.5). The top left image shows the selection of a reference subset. The subset is then padded in the picture in the bottom left and is fed as an input along with the current image to a normalized cross correlation function. The output, on the bottom right, is an array of correlation coefficient values. The red arrow points to the highest value (shown as a white peak); because of the way the subset is constructed and padded in this example, this point refers to the location of the top-left corner in the current configuration. Lastly, the location of the subset is shown with respect to the current configuration.

There are multiple methods for obtaining an initial guess for the nonlinear iterative optimization scheme. The most common and well established method is fast normalized cross correlation (NCC) [4]. This method uses the correlation coefficient in eq.5 and calculates it at every pixel location in the current configuration in a computationally efficient manner. It utilizes the FFT to calculate the non precomputable parts of the numerator, and then utilizes a summed area table to calculate the non precomputable parts of the denominator. The FFT can be used by obtaining the grayscale values within the reference subset and then padding these values with zeros to the same size as the current image (see bottom left of fig.6). At the end of the process, an array of correlation coefficients is obtained; the coordinates of the highest correlation coefficient value will yield the location of the subset in the current configuration. It is important to note that the NCC method can only determine integer displacements (i.e. integer u and v) and therefore cannot provide initial guesses for the other four parameters in **p**. Thus, the normalized cross correlation method is best suited for providing initial guesses in areas of relatively low deformation (i.e. no large rotations or strains). The process is outlined in fig.6; a more detailed and excellent treatment on the implementation of NCC is available at [4].

Note that NCC is not the only method available to provide an initial guess. More recently, Bing Pan has introduced a SIFT based algorithm [5] which can provide an initial guess for all 6 deformation parameters by forming correspondences between SIFT feature points in the reference and current configuration. The feature points which lie within a selected reference region, and their corresponding locations in the current configuration, can be formulated into an over-constrained linear system and then solved for using a linear least squares solver.

3.2 - Gauss-Newton nonlinear iterative least squares method

The Gauss-Newton method is used to find the roots of a function in the case that an analytic solution is not available. This scenario can be extrapolated to optimization by finding the roots of the derivative of a function. Even further, it can be generalized to multivariate optimization by using the gradient in place of the derivative, and then determining where the norm of the gradient converges to zero.

The function, C_{LS} (eq.6), used in the proceeding FA-GN and IC-GN sections, is defined here as a function that accepts a single argument, **p**, which can be applied as either $\mathbf{p}_{rr}$ or $\mathbf{p}_{rc}$. If the argument is applied as $\mathbf{p}_{rr}$, then it is assumed $\mathbf{p}_{rc}$ has already been specified and is a constant; on the other hand, if the argument is applied as $\mathbf{p}_{rc}$, then it is assumed $\mathbf{p}_{rr}$ has been applied and is a constant. How the argument is applied depends on which method (FA-GN or IC-GN) is being used, and will be made clear in the respective sections.

Anyway, the general form of the iterative equation used in this analysis can be derived by taking the second order Taylor series expansion of C_{LS} around $\mathbf{p}_o$, where $\mathbf{p}_o$ contains the starting deformation parameters (i.e. some form of an initial guess), and then determining where its derivative with respect to $\Delta\mathbf{p}$ is equal to the zero vector.

$C_{LS}(\boldsymbol{p}_o + \Delta\boldsymbol{p}) \approx C_{LS}(\boldsymbol{p}_o) + \nabla C_{LS}(\boldsymbol{p}_o)^{T} * \Delta\boldsymbol{p} + \frac{1}{2} * \Delta\boldsymbol{p}^{T} * \nabla\nabla C_{LS}(\boldsymbol{p}_o) * \Delta\boldsymbol{p}$	Eq.9
$\frac{dC_{LS}(\boldsymbol{p}_o + \Delta\boldsymbol{p})}{d\Delta\boldsymbol{p}} \approx \nabla C_{LS}(\boldsymbol{p}_o) + \nabla\nabla C_{LS}(\boldsymbol{p}_o) * \Delta\boldsymbol{p} = \boldsymbol{0}$	Eq.10

Where $\nabla C_{LS}(\mathbf{p}_o)$ is the gradient of C_{LS} at $\mathbf{p}_o$ and $\nabla\nabla C_{LS}(\mathbf{p}_o)$ is the hessian of C_{LS} at $\mathbf{p}_o$. The basic idea behind this is that the Taylor series allows us to form an approximation to the C_{LS} function around $\mathbf{p}_o$ with another function whose minimum we can find analytically (by taking it's derivative and explicitly solving for $\Delta\mathbf{p}$). This will allow us to solve for $\mathbf{p}_o + \Delta\mathbf{p}$, which should be closer to the solution. We can then iterate until an appropriately close solution is found. The general form of the iterative optimization equation used in this analysis is rearranged from eq.10 and shown below:

$\nabla\nabla C_{LS}(\boldsymbol{p}_o) * \Delta\boldsymbol{p} = -\nabla C_{LS}(\boldsymbol{p}_o)$	Eq.11

Eq.11 is actually the Newton-Raphson iterative scheme. It becomes the Gauss-Newton iterative scheme by using an approximation to the hessian matrix, which is elaborated on in the specific FA-GN and IC-GN sections.

3.3 - Forward additive method

The FA-GN method is essentially the standard application of a Gauss-Newton iterative solver. In this situation, the final reference subset location remains constant and $\mathbf{p}_{rr}$ is set to **0** for every iteration. On the other hand, the final current subset location is allowed to deform and $\mathbf{p}_{rc}$ is set to $\mathbf{p}_{old}$ at the beginning of every iteration, where $\mathbf{p}_{old}$ is the deformation parameters found from the previous iteration, using the forward additive technique (eq.19), or the initial guess. This means $\mathbf{p}_o$ from eq.11 equals $\mathbf{p}_{old}$.

The compact form of C_{LS} for this method is:

$C_{LS}(\boldsymbol{p}_{old} + \Delta\boldsymbol{p}) = \sum\left[\frac{f\left(\xi_{\mathrm{ref}_c} + w(\Delta\xi_{ref}; \boldsymbol{0})\right) - f_m}{\sqrt{\sum\left[f\left(\xi_{\mathrm{ref}_c} + w(\Delta\xi_{ref}; \boldsymbol{0})\right) - f_m\right]^2}} - \frac{g\left(\xi_{\mathrm{ref}_c} + w(\Delta\xi_{ref}; \boldsymbol{p}_{old} + \Delta\boldsymbol{p})\right) - g_m}{\sqrt{\sum\left[g\left(\xi_{\mathrm{ref}_c} + w(\Delta\xi_{ref}; \boldsymbol{p}_{old} + \Delta\boldsymbol{p})\right) - g_m\right]^2}}\right]^2$	Eq.12

Where $f\left(\xi_{\mathrm{ref}_c} + w(\Delta\xi_{ref}; \boldsymbol{0})\right)$ represents a vector of sampled reference image grayscale values and $g\left(\xi_{\mathrm{ref}_c} + w(\Delta\xi_{ref}; \boldsymbol{p}_{old} + \Delta\boldsymbol{p})\right)$ represents a vector of sampled current image grayscale values.

Also, for simplicity, we assume that :

$\frac{d}{d\Delta\boldsymbol{p}}(g_m) \approx 0$	Eq.13
$\frac{d}{d\Delta\boldsymbol{p}}\left(\sqrt{\sum\left[g\left(\xi_{\mathrm{ref}_c} + w\left(\Delta\xi_{ref}; \boldsymbol{p}_{old}\right)\right) - g_m\right]^2}\right) \approx 0$	Eq.14

Which are reasonable assumptions if the average grayscale values of the subset doesn't vary much as the subset is deformed by **p**; these assumptions have also been verified empirically to be acceptable. The gradient for this case is found to be:

	Eq.15

$$\nabla C_{LS}(\mathbf{p}_{old}) = \frac{dC_{LS}(\mathbf{p}_{old})}{d\Delta\mathbf{p}} \approx \frac{-2}{\sqrt{\sum\left[g\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{p}_{old}\right)\right)-g_m\right]^2}}\sum\left[\left[\frac{f\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{0}\right)\right)-f_m}{\sqrt{\sum\left[f\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{0}\right)\right)-f_m\right]^2}} - \frac{g\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{p}_{old}\right)\right)-g_m}{\sqrt{\sum\left[g\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{p}_{old}\right)\right)-g_m\right]^2}}\right]\left[\frac{d}{d\Delta\mathbf{p}}g\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{p}_{old}\right)\right)\right]\right]$$

The Hessian is:

$$\nabla\nabla C_{LS}(\mathbf{p}_{old}) = \frac{d^2C_{LS}(\mathbf{p}_{old})}{d\Delta\mathbf{p}^2} \approx \frac{-2}{\sqrt{\sum\left[g\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{p}_{old}\right)\right)-g_m\right]^2}}\left\{\sum\left[-\frac{\frac{d}{d\Delta\mathbf{p}}g\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{p}_{old}\right)\right)}{\sqrt{\sum\left[g\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{p}_{old}\right)\right)-g_m\right]^2}}\right]\left[\frac{d}{d\Delta\mathbf{p}}g\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{p}_{old}\right)\right)\right]^T + \sum\left[\frac{f\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{0}\right)\right)-f_m}{\sqrt{\sum\left[f\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{0}\right)\right)-f_m\right]^2}} - \frac{g\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{p}_{old}\right)\right)-g_m}{\sqrt{\sum\left[g\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{p}_{old}\right)\right)-g_m\right]^2}}\right]\left[\frac{d^2}{d\Delta\mathbf{p}^2}g\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{p}_{old}\right)\right)\right]\right\} \quad \text{Eq.16}$$

This is the hessian for the Newton-Raphson iterations. It becomes the Gauss-Newton hessian with the following assumption:

$$\sum\left[\frac{f\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{0}\right)\right)-f_m}{\sqrt{\sum\left[f\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{0}\right)\right)-f_m\right]^2}} - \frac{g\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{p}_{old}\right)\right)-g_m}{\sqrt{\sum\left[g\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{p}_{old}\right)\right)-g_m\right]^2}}\right]\left[\frac{d^2}{d\Delta\mathbf{p}^2}g\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{p}_{old}\right)\right)\right] \approx 0 \quad \text{Eq.17}$$

This is a reasonable assumption if $\mathbf{p}_{old}$ is close to the optimal solution (this makes the first quantity in brackets close to zero) and also if the second derivatives of the grayscale values are close to zero (this depends on the image and interpolation used, but in general the grayscale values should be pretty smooth so this quantity should be small). The two small quantities that multiply each other in eq.17 result in a quantity small in magnitude.

The above assumption does not come without a cost. The Gauss-Newton method actually takes more iterations to converge than the standard Newton-Raphson method. However, a single iteration of the Gauss-Newton iteration is cheaper than a single Newton-Raphson iteration. So, a speed improvement will only be realized if the cost of the Gauss-Newton iterations are cheap enough to offset the added cost of more iterations. In my experience, this is the case for smaller subsets, although for larger subsets the Newton-Raphson method becomes faster (note that, from my experience, this does not seem to be the case for the inverse compositional method though). Outside of computational speed, another reason to implement the Gauss-Newton method is that it actually has much better convergence characteristics then the Newton-Raphson method. It is also simpler to implement than the Newton-Raphson method.

Anyway, this yields a final form of the hessian as:

$$\nabla\nabla C_{LS}(\mathbf{p}_{old}) = \frac{d^2C_{LS}(\mathbf{p}_{old})}{\Delta\mathbf{p}} \approx \frac{2}{\sum\left[g\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{p}_{old}\right)\right)-g_m\right]^2}\sum_{(i,j)\in S}\left[\frac{\partial}{\partial\Delta\mathbf{p}}g\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{p}_{old}\right)\right)\right]\left[\frac{\partial}{\partial\Delta\mathbf{p}}g\left(\xi_{ref_c}+w\left(\Delta\xi_{ref};\mathbf{p}_{old}\right)\right)\right]^T \quad \text{Eq.18}$$

Note that for the gradient and hessian to be calculable, interpolation is needed (described in sections 3.5-3.7), and this interpolation procedure happens to be one of the most expensive parts of the computational process, especially if high order interpolation is used.

The general method of the FA-GN iterations is to calculate the gradient and hessian, and then calculate Δ**p** using eq.11,eq.15, and eq.18 with Cholesky decomposition, since the Gauss-Newton hessian should be symmetric positive definite. This increment, Δ**p**, is then used to update **p**$_{old}$ in the following way:

$$p_{new} = p_{old} + \Delta p \qquad \text{Eq.19}$$

where **p**$_{old}$ is set to **p**$_{new}$ at the beginning of every iteration. This method is called the forward additive method because of how **p**$_{old}$ is updated as shown above (it will make more sense as to why we make this distinction once the IC-GN method is discussed). It's important to note that the most expensive part of the calculations are the evaluations of g and dg/dΔ**p** at $\xi_{ref_c} + w(\Delta\xi_{ref}; p_{old})$. It turns out that through the properties of affine transformations and a clever trick implemented by [6], the iterative procedure can be reformulated to be much more efficient, which is explained in further detail in the IC-GN method section.

Anyway, the overall flow of the FA-GN method is given in fig.7 below:

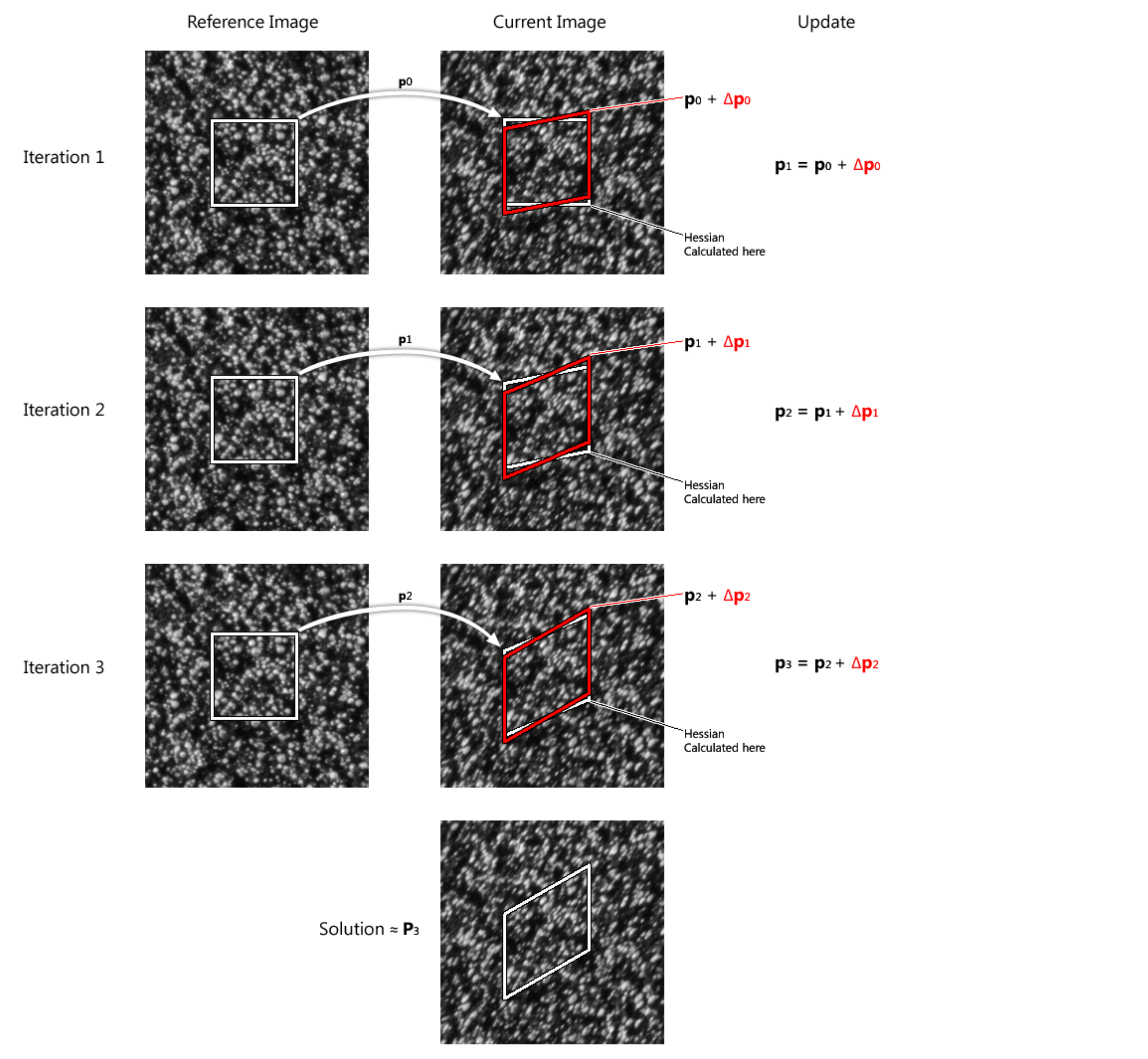


Fig.7. This is a graphical representation of the FA-GN method for image alignment. The reference image has a shear applied to it. The most important thing to note here is where the hessian is evaluated at $\xi_{ref_c} + w(\Delta\xi_{ref}; p_{old})$ at the current image for every iteration). This quantity is the computational bottleneck of the algorithm and provides good insight to understanding the effectiveness of the IC-GN method.

3.4 - Inverse compositional method

In this method, a couple things are changed from the FA-GN method. First of all, the final reference subset location is allowed to vary, but **p**$_{rr}$ is set to **0** at the beginning of every iteration. This means **p**$_{o}$ from eq.11 equals **0**. On the other hand, the final current subset location is not allowed to vary, but is set to **p**$_{old}$ at the beginning of every iteration. **p**$_{old}$ is the updated parameters for **p**$_{rc}$ found from the previous iteration using the inverse compositional technique (eq.28), which is explained later. The motivation for making this switch is that the hessian is calculated using the reference subset which begins at the same location at every iteration, meaning the hessian is the same for every iteration and therefore only needs to be calculated once. This idea will be made more clear in the following paragraphs.

The compact form of C_{LS} for this method is:

$$C_{LS}(\Delta \boldsymbol{p}) = \sum\left[\frac{f\left(\xi_{\text{ref}_c}+w(\Delta\xi_{\boldsymbol{ref}};\Delta \boldsymbol{p})\right)-f_m}{\sqrt{\sum\left[f\left(\xi_{\text{ref}_c}+w(\Delta\xi_{\boldsymbol{ref}};\Delta \boldsymbol{p})\right)-f_m\right]^2}} - \frac{g\left(\xi_{\text{ref}_c}+w(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{p}_{\boldsymbol{old}})\right)-g_m}{\sqrt{\sum\left[g\left(\xi_{\text{ref}_c}+w(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{p}_{\boldsymbol{old}})\right)-g_m\right]^2}}\right]^2$$

Eq.20

The assumptions about the averages and the quantities at the bottom remain the same:

$$\frac{d}{d\Delta \boldsymbol{p}}\left(f_m\right) \approx 0$$

Eq.21

$$\frac{d}{d\Delta \boldsymbol{p}}\left(\sqrt{\sum\left[f\left(\xi_{\text{ref}_c}+w\left(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{0}\right)\right)-f_m\right]^2}\right) \approx 0$$

Eq.22

The gradient for this case is found to be:

$$\begin{aligned}\nabla C_{LS}(\boldsymbol{0}) &= \frac{dC_{LS}(\boldsymbol{0})}{d\Delta \boldsymbol{p}}\\ &\approx \frac{2}{\sqrt{\sum\left[f\left(\xi_{\text{ref}_c}+w\left(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{0}\right)\right)-f_m\right]^2}}\sum\Bigg[\Bigg[\frac{f\left(\xi_{\text{ref}_c}+w\left(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{0}\right)\right)-f_m}{\sqrt{\sum\left[f\left(\xi_{\text{ref}_c}+w\left(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{0}\right)\right)-f_m\right]^2}}\\ &\quad -\frac{g\left(\xi_{\text{ref}_c}+w\left(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{p}_{\boldsymbol{old}}\right)\right)-g_m}{\sqrt{\sum\left[g\left(\xi_{\text{ref}_c}+w\left(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{p}_{\boldsymbol{old}}\right)\right)-g_m\right]^2}}\Bigg]\left[\frac{d}{d\Delta \boldsymbol{p}}f\left(\xi_{\text{ref}_c}+w\left(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{0}\right)\right)\right]\Bigg]\end{aligned}$$

Eq.23

The Hessian is:

$$\begin{aligned}\nabla\nabla C_{LS}(\boldsymbol{0}) &= \frac{d^2C_{LS}(\boldsymbol{0})}{d\Delta \boldsymbol{p}^2}\\ &\approx \frac{2}{\sqrt{\sum\left[f\left(\xi_{\text{ref}_c}+w(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{0})\right)-f_m\right]^2}}\left\{\sum\left[\frac{\frac{d}{d\Delta \boldsymbol{p}}f\left(\xi_{\text{ref}_c}+w(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{0})\right)}{\sqrt{\sum\left[f\left(\xi_{\text{ref}_c}+w(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{0})\right)-f_m\right]^2}}\right]\left[\frac{d}{d\Delta \boldsymbol{p}}f\left(\xi_{\text{ref}_c}\right.\right.\right.\\ &\quad \left.\left.+w(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{0})\right)\right]^T\\ &\quad +\sum\left[\frac{f\left(\xi_{\text{ref}_c}+w(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{0})\right)-f_m}{\sqrt{\sum\left[f\left(\xi_{\text{ref}_c}+w(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{0})\right)-f_m\right]^2}}\right.\\ &\quad \left.\left.-\frac{g\left(\xi_{\text{ref}_c}+w(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{p}_{\boldsymbol{old}})\right)-g_m}{\sqrt{\sum\left[g\left(\xi_{\text{ref}_c}+w(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{p}_{\boldsymbol{old}})\right)-g_m\right]^2}}\right]\left[\frac{d^2}{d\Delta \boldsymbol{p}^2}f\left(\xi_{\text{ref}_c}+w(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{0})\right)\right]\right\}\end{aligned}$$

Eq.24

Once again, the Gauss-Newton assumption is:

$$\begin{aligned}&\sum\left[\frac{f\left(\xi_{\text{ref}_c}+w\left(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{0}\right)\right)-f_m}{\sqrt{\sum\left[f\left(\xi_{\text{ref}_c}+w\left(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{0}\right)\right)-f_m\right]^2}}-\frac{g\left(\xi_{\text{ref}_c}+w\left(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{p}_{\boldsymbol{old}}\right)\right)-g_m}{\sqrt{\sum\left[g\left(\xi_{\text{ref}_c}+w\left(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{p}_{\boldsymbol{old}}\right)\right)-g_m\right]^2}}\right]\left[\frac{d^2}{d\Delta \boldsymbol{p}^2}f\left(\xi_{\text{ref}_c}+w\left(\Delta\xi_{\boldsymbol{ref}};\boldsymbol{0}\right)\right)\right]\\ &\quad\approx 0\end{aligned}$$

Eq.25

This yields the hessian in the final form:

$\nabla\nabla C_{LS}(\mathbf{0}) \approx \frac{dC_{LS}(\mathbf{0})}{d\Delta\boldsymbol{p}^2} \approx \frac{2}{\sum\left[f\left(\xi_{\mathrm{ref}_c} + w(\Delta\xi_{ref};\mathbf{0})\right) - f_m\right]^2} \sum\left[\frac{d}{d\Delta\boldsymbol{p}} f\left(\xi_{\mathrm{ref}_c} + w(\Delta\xi_{ref};\mathbf{0})\right)\right]\left[\frac{d}{d\Delta\boldsymbol{p}} f\left(\xi_{\mathrm{ref}_c} + w(\Delta\xi_{ref};\mathbf{0})\right)\right]^T$	Eq.26

Δ**p** can be solved for using eq.11, eq.23 and eq.26 with Cholesky decomposition. The next step is to figure out what to do with Δ**p**. It turns out that a closer approximation to $\mathbf{p}_{\mathrm{rc}}^{*}$ can be found by taking $\mathbf{p}_{\mathrm{rc}}$ ($\mathbf{p}_{\mathrm{old}}$ in this case) and composing it with the inverse of $\mathbf{p}_{\mathrm{rr}}$ ($\Delta\mathbf{p}$ in this case). The act of doing this determines the direct transformation from the final reference subset to the final current subset. A graphical figure is shown below:

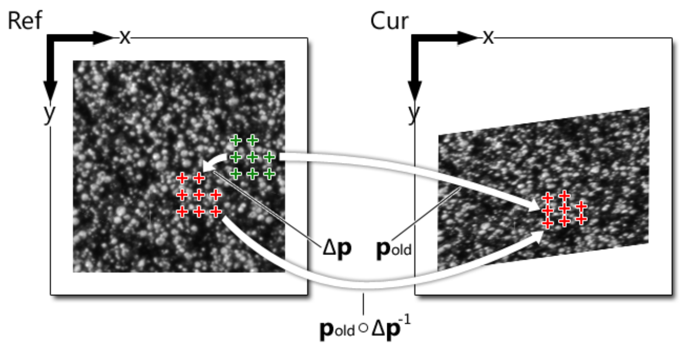


Fig.8. This a graphical representation of the inverse compositional update. $\mathbf{p}_{\mathrm{rr}}$ is equal to $\Delta\mathbf{p}$ and $\mathbf{p}_{\mathrm{rc}}$ is equal to $\mathbf{p}_{\mathrm{old}}$. As long as the translational component of $\Delta\mathbf{p}$ is small in magnitude, the composition of $\mathbf{p}_{\mathrm{old}}$ and the inverse of $\Delta\mathbf{p}$ should be closer to $\mathbf{p}_{\mathrm{rc}}^{*}$ than $\mathbf{p}_{\mathrm{old}}$.

This update can be shown in matrix form as shown below:

$\begin{bmatrix} 1+\frac{du}{dx}_{new} & \frac{du}{dy}_{new} & u_{new} \\ \frac{dv}{dx}_{new} & 1+\frac{dv}{dy}_{new} & v_{new} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1+\frac{du}{dx}_{old} & \frac{du}{dy}_{old} & u_{old} \\ \frac{dv}{dx}_{old} & 1+\frac{dv}{dy}_{old} & v_{old} \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1+\Delta\frac{du}{dx} & \Delta\frac{du}{dy} & \Delta u \\ \Delta\frac{dv}{dx} & 1+\Delta\frac{dv}{dy} & \Delta v \\ 0 & 0 & 1 \end{bmatrix}^{-1}$	Eq.27
$w\left(\Delta\xi_{ref};\mathbf{p}_{\mathrm{new}}\right) = w\left(w\left(\Delta\xi_{ref};\Delta\mathbf{p}\right)^{-1};\mathbf{p}_{\mathrm{old}}\right)$	Eq.28

where $\mathbf{p}_{\mathrm{old}}$ is set to $\mathbf{p}_{\mathrm{new}}$ at the beginning of every iteration. The inverse compositional method gets its name from how $\mathbf{p}_{\mathrm{old}}$ is updated as shown in eq.27. Overall, it is important to reiterate that starting the final reference subset at a deformation of **0** at the beginning of each iteration allows us to calculate the hessian at $\xi_{\mathrm{ref}_c} + w(\Delta\xi_{ref};\mathbf{0})$ for each iteration, which means it only needs to be calculated once.

This method confused me when I first read about it, so I provided some examples below to build better intuition for it:

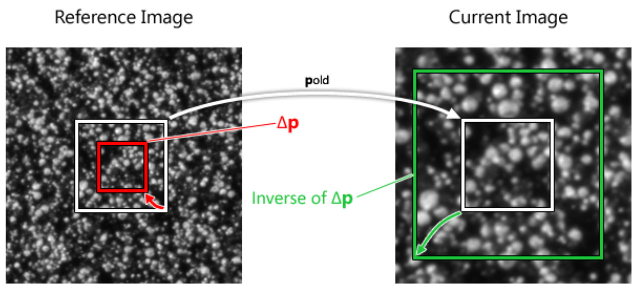


Fig.9. This figure demonstrates what needs to be done with the $\Delta\mathbf{p}$ quantity after it has been calculated. The above figure should give some intuition on how this quantity is used to calculate $\mathbf{p}_{\mathrm{new}}$.

Assume that in fig.9, we have been provided with an initial guess of $\mathbf{p}_{\mathrm{old}}$ and that we have run one iteration of the IC-GN algorithm and have found $\Delta\mathbf{p}$. Note that the current image is the reference image enlarged by 100% in both dimensions. Assume that $\mathbf{p}_{\mathrm{old}}$ is an initial guess only using translation and that an answer very close to the solution has been found in a single iteration. This means that $\Delta\mathbf{p}$ should act to reduce the dimensions of the initial reference subset by about 50% (shown as the red square on the left side of fig.9). To apply the inverse composition update correctly (eq.27), we see that the inverse of $\Delta\mathbf{p}$ must be applied, followed by applying $\mathbf{p}_{\mathrm{old}}$. You can see the appropriateness of doing this, as applying the inverse of Δ**p** will enlarge the subset dimensions 100%, and then $\mathbf{p}_{old}$ will translate it into place; this is shown as the green box on the right side of fig.9. By comparing the white box on the left side of fig.9 (the initial reference subset) with the green box on the right side of fig.9 (the final current subset), we see a good match has been obtained, meaning that $\mathbf{p}_{rc}^*$ has been found.

To reinforce the concept, another example is given below:

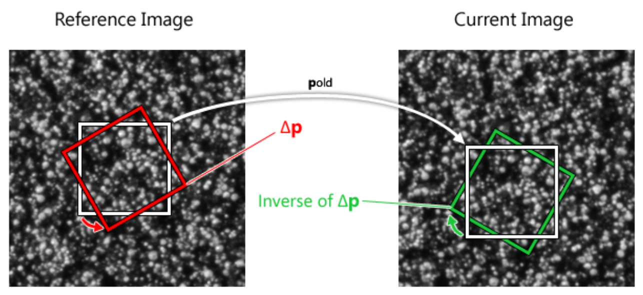


Fig.10. This figure is meant to supplement fig.9 in building intuition for the inverse compositional method.

The example in fig.10 involves a rotation of 30 degrees clockwise from the reference to the current image. Assume $\mathbf{p}_{old}$ is an initial guess involving a translation and also assume that an answer has been found very close to the solution after one iteration. Like the previous example, the inverse transformation of Δ**p** (a rotation in the opposite direction of Δ**p**) is needed to update $\mathbf{p}_{old}$ correctly through eq.27. Once again, comparing the white box on the left side of fig.10 (the initial reference subset) with the green box on the right side of fig.10 (the final current subset), we note that a good match is obtained, meaning that $\mathbf{p}_{rc}^*$ has once again been found.

Hopefully the above two examples helped build intuition for the IC-GN method. The overall flow of the IC-GN method is shown below:

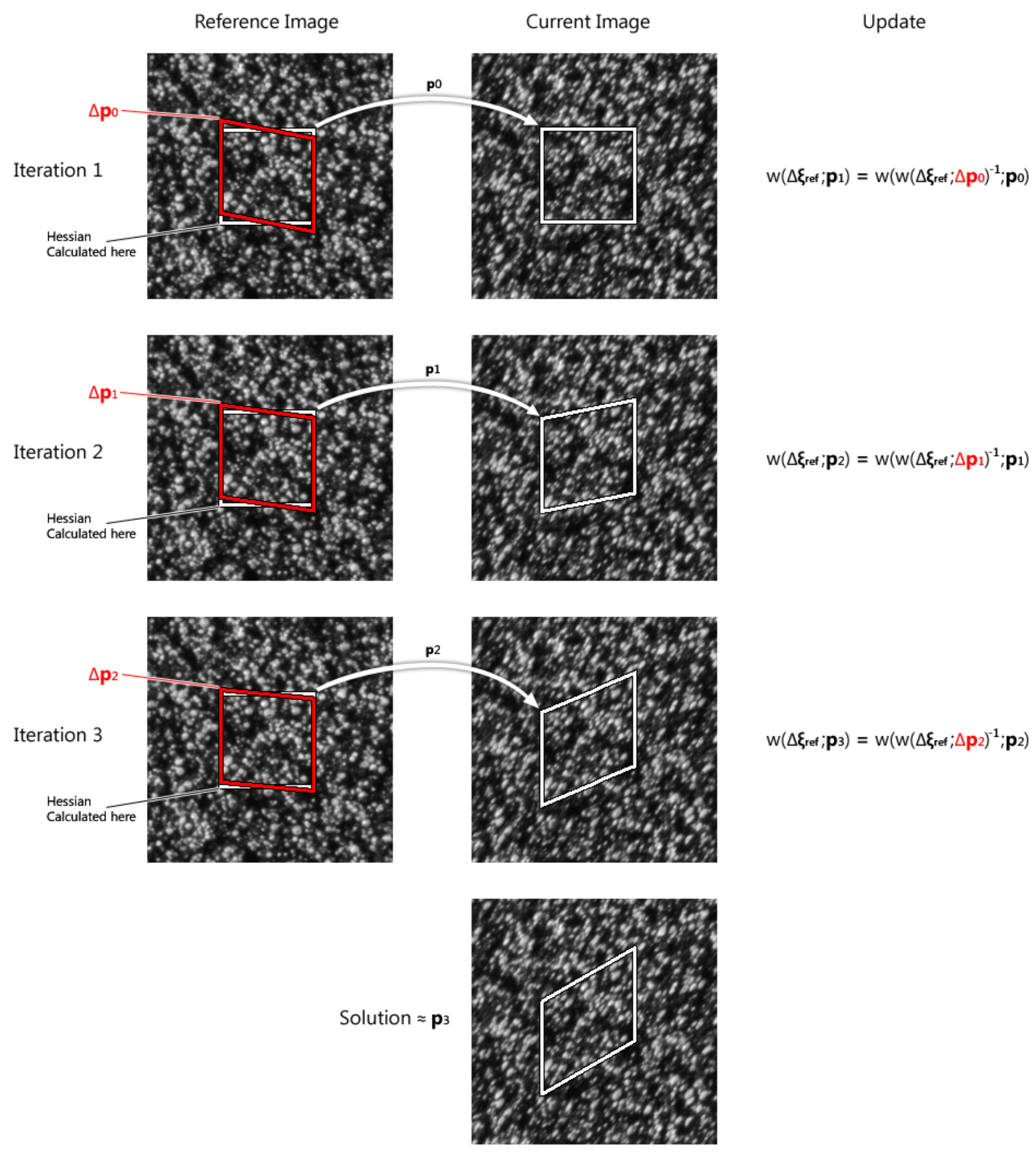


Fig.11. This is a graphical representation of the IC-GN method for image alignment. The reference image has a shear applied to it. The most important thing to note here is where the hessian is evaluated at ($\xi_{ref_c} + w(\Delta\xi_{ref};\mathbf{0})$ at the reference image for every iteration). Because this quantity is evaluated at the same position, it only needs to be computed once. It is highly recommended to compare this figure with fig.7 to have a nice visual comparison between the two iterative methods.

3.5 - Computation of the gradient and hessian for the IC-GN method

Now that the basic overview of the IC-GN method has been given, this write up will proceed by explaining how to specifically compute the gradient and hessian quantities for the IC-GN process. The quantity $f\left(\xi_{ref_c} + w\left(\Delta\xi_{ref};\mathbf{0}\right)\right)$ isn't a problem, because the initial reference subset is picked so that $\Delta\xi_{ref}$ and ξ_{ref_c} lie on integer pixel locations, so the aforementioned quantity can be directly obtained from the reference grayscale values. The quantity $\frac{d}{d\Delta p} f\left(\xi_{ref_c} + w\left(\Delta\xi_{ref};\mathbf{0}\right)\right)$, on the other hand, requires a little work. It needs to be expanded with the chain rule in order to be computable. For convenience, I've rewritten eq.4 as shown below:

$\tilde{x}_{ref_i} = x_{ref_i} + \Delta u + \Delta\frac{\partial u}{\partial x}\left(x_{ref_i} - x_{ref_c}\right) + \Delta\frac{\partial u}{\partial y}\left(y_{ref_j} - y_{ref_c}\right)$ $\tilde{y}_{ref_j} = y_{ref_j} + \Delta v + \Delta\frac{\partial v}{\partial x}\left(x_{ref_i} - x_{ref_c}\right) + \Delta\frac{\partial v}{\partial y}\left(y_{ref_j} - y_{ref_c}\right)$	$(i,j) \in S$	Eq.29

Using the chain rule on $\frac{d}{d\Delta p} f\left(\xi_{ref_c} + w\left(\Delta\xi_{ref};\mathbf{0}\right)\right)$, we obtain:

$\frac{d}{d\Delta\mathbf{p}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) = \frac{\partial}{\partial \tilde{x}_{ref_i}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) * \frac{d\tilde{x}_{ref_i}}{d\Delta\mathbf{p}} + \frac{\partial}{\partial \tilde{y}_{ref_j}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) * \frac{d\tilde{y}_{ref_j}}{d\Delta\mathbf{p}}$	Eq.30

For convenience, I've expanded out these quantities as shown below:

$\frac{\partial}{\partial \Delta u} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) = \frac{\partial}{\partial \tilde{x}_{ref_i}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right)$	Eq.31
$\frac{\partial}{\partial \Delta v} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) = \frac{\partial}{\partial \tilde{y}_{ref_j}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right)$	Eq.32
$\frac{\partial}{\partial\left(\Delta\frac{\partial u}{\partial x}\right)} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) = \frac{\partial}{\partial \tilde{x}_{ref_i}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) * (x_{ref_i} - x_{ref_c})$	Eq.33
$\frac{\partial}{\partial(\Delta\frac{\partial u}{\partial y})} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) = \frac{\partial}{\partial \tilde{x}_{ref_i}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) * (y_{ref_j} - y_{ref_c})$	Eq.34
$\frac{\partial}{\partial(\Delta\frac{\partial v}{\partial x})} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) = \frac{\partial}{\partial \tilde{y}_{ref_j}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) * (x_{ref_i} - x_{ref_c})$	Eq.35
$\frac{\partial}{\partial(\Delta\frac{\partial v}{\partial y})} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) = \frac{\partial}{\partial \tilde{y}_{ref_j}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) * (y_{ref_j} - y_{ref_c})$	Eq.36

The only two quantities we need to compute for eq.31-36 are $\frac{\partial}{\partial \tilde{x}_{ref_i}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right)$ and $\frac{\partial}{\partial \tilde{y}_{ref_j}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right)$. These can be computed in various ways, but in Ncorr, biquintic B-spline interpolation is used, so in order to keep it consistent I've decided to compute these as if they are the partial derivatives of biquintic B-splines, which will be explained in the interpolation section.

The last quantity we need to address is $g\left(\xi_{ref_c} + w\left(\Delta\xi_{ref}; \mathbf{p}_{old}\right)\right)$, which requires interpolation. Once $\frac{\partial}{\partial \tilde{x}_{ref_i}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right)$ and $\frac{\partial}{\partial \tilde{y}_{ref_j}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right)$ are precomputed for the entire reference image, and $g\left(\xi_{ref_c} + w\left(\Delta\xi_{ref}; \mathbf{p}_{old}\right)\right)$ is computable, then eq.23 and eq.26 can be computed and iterated with eq.11 to find a close approximation to $\mathbf{p}_{rc}^{*}$. The final step for the iterative solver is explaining the interpolation process, which is done in the next section.

3.6 - Biquintic B-spline background theory

The purpose of this section is to discuss biquintic interpolation in order to calculate $\frac{\partial}{\partial \tilde{x}_{ref_i}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right)$, $\frac{\partial}{\partial \tilde{y}_{ref_j}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right)$, and $g\left(\tilde{x}_{cur_i}, \tilde{y}_{cur_j}\right)$. Biquintic interpolation is used because it mitigates DIC errors which result from interpolating signals with high frequency content [7]. But, it is computationally expensive to implement, although I justify its use because DIC is often done "off-line" (i.e. after experiments are completed). This section will mainly focus on the implementation of biquintic B-splines with a little bit of background theory; if a more detailed explanation/treatment is desired then the reader is directed to [8] and [9].

The main idea behind B-spline interpolation is to approximate the image grayscale surface with a linear combination of B-spline basis "splines." These splines are essentially small "bumps" in 2D space with finite support (meaning only a small portion of their range is nonzero). These splines are scaled via the B-spline coefficients and then a linear combination of these scaled splines form an approximation of the surface. Once this approximation is complete, points can be interpolated through convolutions, which reduces to a series of simple dot products.

Note that all of the theory given in the next sections are for the 1D case. B-splines have a very nice property in that they are separable [8]; this means that 2D interpolations can be broken down into a series of 1D interpolations which is computationally faster than a single 2D interpolation (this idea will be expanded upon later). Anyway, the interpolation can be expressed as:

$g(x) = \sum_{k \in Z} c(k)\beta^n(x-k)$	Eq.37

where c(k),β^n(x-k), and g(x) are the B-spline coefficent value at integer k, the B-spline kernel value at x-k, and the interpolated signal value at x, respectively. n is the B-spline kernel order, which I set to 5 (the quintic kernel) and Z is the set of integers. Note that the B-spline coefficients are not equivalent to the data samples (unlike in other forms of interpolation - i.e. bicubic keys[10]), and thus must be solved for directly. The equation for the B-spline kernel is:

$\beta^n(x) = \frac{1}{n!}\sum_{k=0}^{n+1}\binom{n+1}{k}(-1)^k\left(x-k+\frac{n+1}{2}\right)_+^n$	Eq.38

When solved for the quintic case, this yields:

$$\beta^5(x) = \begin{cases} \frac{1}{120}*x^5+\frac{1}{8}*x^4+\frac{3}{4}*x^3+\frac{9}{4}*x^2+\frac{27}{8}*x+\frac{81}{40} & -2 \geq x \geq -3 \\ -\frac{1}{24}*x^5-\frac{3}{8}*x^4-\frac{5}{4}*x^3-\frac{7}{4}*x^2-\frac{5}{8}*x+\frac{17}{40} & -1 \geq x \geq -2 \\ \frac{1}{12}*x^5+\frac{1}{4}*x^4-\frac{1}{2}*x^2+\frac{11}{20} & 0 \geq x \geq -1 \\ -\frac{1}{12}*x^5+\frac{1}{4}*x^4-\frac{1}{2}*x^2+\frac{11}{20} & 1 \geq x \geq 0 \\ \frac{1}{24}*x^5-\frac{3}{8}*x^4+\frac{5}{4}*x^3-\frac{7}{4}*x^2+\frac{5}{8}*x+\frac{17}{40} & 2 \geq x \geq 1 \\ -\frac{1}{120}*x^5+\frac{1}{8}*x^4-\frac{3}{4}*x^3+\frac{9}{4}*x^2-\frac{27}{8}*x+\frac{81}{40} & 3 \geq x \geq 2 \end{cases} \quad \text{Eq.39}$$

Now we have most of the information needed to begin the interpolation process. The first step is to determine the B-spline coefficients. They can be found by using deconvolution. Looking at eq.37 and taking the DFT, we find:

$$F\{g\} = F\{c\} * F\{\beta^n\} \quad \text{Eq.40}$$

The goal is to solve for c, the B-spline coefficients. This can be done by dividing the Fourier coefficients of the signal element-wise with the Fourier coefficients of the B-spline kernel as shown below:

$$F\{c\} = \frac{F\{g\}}{F\{\beta^n\}} \quad \text{Eq.41}$$

Taking the inverse DFT of eq.41 will then yield the B-spline coefficients, although caution should be exercised when using this method due to the circular nature of the DFT. To mitigate wrap-around errors, padding should be used.

After obtaining the B-spline coefficients, the image array can be interpolated point-wise by using eq.37. This is carried out by taking a series of dot products with the columns of the B-spline coefficient array and B-spline kernel, and then taking a single dot product across the resulting row of interpolated values (note that the order of this operation doesn't matter). The first step of the aforementioned process can be thought of as interpolating the 2D B-spline grid to obtain 1D B-spline coefficient values, and then the second step as interpolating the grayscale value from these 1D B-spline coefficient values. This is expanded upon in section 3.7.

3.7 - Biquintic B-spline implementation

The steps for obtaining the B-spline coefficients are outlined below:

1. Make a copy of the grayscale array and pad it (any method can be used; I use the border values to expand the data as shown in the top row of fig.13). Then, sample the B-spline kernel at -2,-1,0,1, 2, and 3 as shown below:

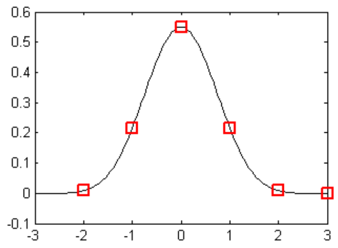


Fig.12. Quintic kernel sampled at integer locations.

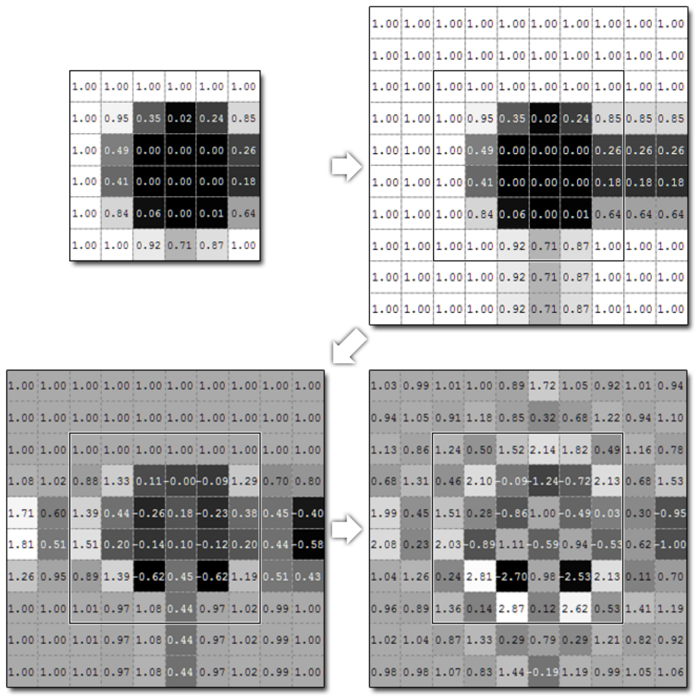
This will form the quintic B-spline vector, $\mathbf{b_o}$:

$$\mathbf{b_o} = \{1/120\ 13/60\ 11/20\ 13/60\ 1/120\ 0\}^{\mathrm{T}} \quad \text{Eq.42}$$

2. Center the kernel, then pad it with zeros to the same size as the number of columns (the width) of the image grayscale array. Take the FFT of the padded kernel, and then store it in place.

3. Take the FFT of an image row, then *divide* the Fourier coefficients of the image row element-wise with the Fourier coefficients from the the padded B-spline. Afterward, take the inverse FFT of the results and store them in place (in the padded grayscale array). Do this for all the image rows. This step is illustrated in the bottom-left picture of fig.13.

4. Repeat steps 2-3, except column-wise, with the array obtained at the end of step 3. The result will be the B-spline coefficients for a biquintic kernel of the image array. This step is shown in the bottom-right picture of fig.13. The overall process is demonstrated in the fig.13.



1.00	1.00	1.00	1.00	1.00	1.00
1.00	0.95	0.35	0.02	0.24	0.85
1.00	0.49	0.00	0.00	0.00	0.26
1.00	0.41	0.00	0.00	0.00	0.18
1.00	0.84	0.06	0.00	0.01	0.64
1.00	1.00	0.92	0.71	0.87	1.00

1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	0.95	0.35	0.02	0.24	0.85	0.85	0.85
1.00	1.00	1.00	0.49	0.00	0.00	0.00	0.26	0.26	0.26
1.00	1.00	1.00	0.41	0.00	0.00	0.00	0.18	0.18	0.18
1.00	1.00	1.00	0.84	0.06	0.00	0.01	0.64	0.64	0.64
1.00	1.00	1.00	1.00	0.92	0.71	0.87	1.00	1.00	1.00
1.00	1.00	1.00	1.00	0.92	0.71	0.87	1.00	1.00	1.00
1.00	1.00	1.00	1.00	0.92	0.71	0.87	1.00	1.00	1.00

1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1.08	1.02	0.88	1.33	0.11	-0.00	-0.09	1.29	0.70	0.80
1.71	0.60	1.39	0.44	-0.26	0.18	-0.23	0.38	0.45	-0.40
1.81	0.51	1.51	0.20	-0.14	0.10	-0.12	0.20	0.44	-0.58
1.26	0.95	0.89	1.39	-0.62	0.45	-0.62	1.19	0.51	0.43
1.00	1.00	1.01	0.97	1.08	0.44	0.97	1.02	0.99	1.00
1.00	1.00	1.01	0.97	1.08	0.44	0.97	1.02	0.99	1.00
1.00	1.00	1.01	0.97	1.08	0.44	0.97	1.02	0.99	1.00

1.03	0.99	1.01	1.00	0.89	1.72	1.05	0.92	1.01	0.94
0.94	1.05	0.91	1.18	0.85	0.32	0.68	1.22	0.94	1.10
1.13	0.86	1.24	0.50	1.52	2.14	1.82	0.49	1.16	0.78
0.68	1.31	0.46	2.10	-0.09	-1.24	-0.72	2.13	0.68	1.53
1.99	0.45	1.51	0.28	-0.86	1.00	-0.49	0.03	0.30	-0.95
2.08	0.23	2.03	-0.89	1.11	-0.59	0.94	-0.53	0.62	-1.00
1.04	1.26	0.24	2.81	-2.70	0.98	-2.53	2.13	0.11	0.70
0.96	0.89	1.36	0.14	2.87	0.12	2.62	0.53	1.41	1.19
1.02	1.04	0.87	1.33	0.29	0.79	0.29	1.21	0.82	0.92
0.98	0.98	1.07	0.83	1.44	-0.19	1.19	0.99	1.05	1.06

Fig.13 The top left is the original array of grayscale values whose B-spline coefficients we want to calculate. The next step is to make a copy of the data and then pad it; in this example the padding parameter is set to 2. Next, deconvolution via FFT is applied to each row. Lastly, deconvolution is applied to each column. The end result on the bottom right are the associated B-spline coefficients for the top left image. Note that in this example we only use two pixels of padding on each side of the image, but in application, more padding should be used, especially if interpolation is done near the boundary (as ringing may occur [8]).

Now that the B-spline coefficients have been obtained, we can interpolate values at sub pixel locations. The steps are outlined below:

1. Pick a subpixel point, ($\tilde{x}_{cur}, \tilde{y}_{cur}$), within the image array to interpolate.

2. Calculate Δx and Δy by:

$$\Delta x = \tilde{x}_{cur} - x_f$$
$$\Delta y = \tilde{y}_{cur} - y_f \qquad \text{Eq.43}$$

Where x_f = floor($\tilde{x}_{cur}$) and y_f = floor($\tilde{y}_{cur}$).

3. Perform the following operation to obtain the interpolated value:

$$g(\tilde{x}_{cur}, \tilde{y}_{cur}) = \begin{bmatrix} 1 & \Delta y & \Delta y^2 & \Delta y^3 & \Delta y^4 & \Delta y^5 \end{bmatrix} * [QK] * [c]_{(x_f-2:x_f+3,\, y_f-2:y_f+3)} * [QK]^T * \begin{bmatrix} 1 \\ \Delta x \\ \Delta x^2 \\ \Delta x^3 \\ \Delta x^4 \\ \Delta x^5 \end{bmatrix} \qquad \text{Eq.44}$$

Where [QK] is:

$$[QK] = \begin{bmatrix} \frac{1}{120} & \frac{13}{60} & \frac{11}{20} & \frac{13}{60} & \frac{1}{120} & 0 \\ -\frac{1}{24} & -\frac{5}{12} & 0 & \frac{5}{12} & \frac{1}{24} & 0 \\ \frac{1}{12} & \frac{1}{6} & -\frac{1}{2} & \frac{1}{6} & \frac{1}{12} & 0 \\ -\frac{1}{12} & \frac{1}{6} & 0 & -\frac{1}{6} & \frac{1}{12} & 0 \\ \frac{1}{24} & -\frac{1}{6} & \frac{1}{4} & -\frac{1}{6} & \frac{1}{24} & 0 \\ -\frac{1}{120} & \frac{1}{24} & -\frac{1}{12} & \frac{1}{12} & -\frac{1}{24} & \frac{1}{120} \end{bmatrix} \quad \text{Eq.45}$$

and $[c]_{(x_f-2:x_f+3,y_f-2:y_f+3)}$ are the B-spline coefficients for g as shown below:

$$[c]_{(x_f-2:x_f+3,y_f-2:y_f+3)} = \begin{bmatrix} c_{(x_f-2,y_f-2)} & c_{(x_f-1,y_f-2)} & c_{(x_f,y_f-2)} & c_{(x_f+1,y_f-2)} & c_{(x_f+2,y_f-2)} & c_{(x_f+3,y_f-2)} \\ c_{(x_f-2,y_f-1)} & c_{(x_f-1,y_f-1)} & c_{(x_f,y_f-1)} & c_{(x_f+1,y_f-1)} & c_{(x_f+2,y_f-1)} & c_{(x_f+3,y_f-1)} \\ c_{(x_f-2,y_f)} & c_{(x_f-1,y_f)} & c_{(x_f,y_f)} & c_{(x_f+1,y_f)} & c_{(x_f+2,y_f)} & c_{(x_f+3,y_f)} \\ c_{(x_f-2,y_f+1)} & c_{(x_f-1,y_f+1)} & c_{(x_f,y_f+1)} & c_{(x_f+1,y_f+1)} & c_{(x_f+2,y_f+1)} & c_{(x_f+3,y_f+1)} \\ c_{(x_f-2,y_f+2)} & c_{(x_f-1,y_f+2)} & c_{(x_f,y_f+2)} & c_{(x_f+1,y_f+2)} & c_{(x_f+2,y_f+2)} & c_{(x_f+3,y_f+2)} \\ c_{(x_f-2,y_f+3)} & c_{(x_f-1,y_f+3)} & c_{(x_f,y_f+3)} & c_{(x_f+1,y_f+3)} & c_{(x_f+2,y_f+3)} & c_{(x_f+3,y_f+3)} \end{bmatrix} \quad \text{Eq.46}$$

The position of these B-spline coefficients within the B-spline array ultimately depend on the amount of padding used. The top right of fig.15 gives an example of the location of the coefficients within the B-spline array for a given x_f and y_f and a padding of 2.

Many things are happening in eq.44 so I'll break down the equation from left to right. The left portion:

$$[1 \quad \Delta y \quad \Delta y^2 \quad \Delta y^3 \quad \Delta y^4 \quad \Delta y^5] * [QK]$$

Is the matrix form of resampling the quintic B-spline kernel with a shift of Δy. The elements of the resulting vector are shown as the red squares in the figure below:

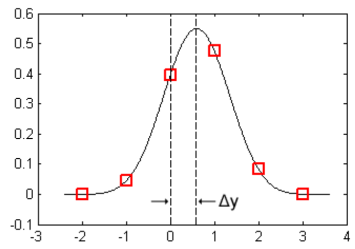


Fig.14 Quintic kernel sampled at non-integer locations.

Right multiplying the portion above by $[c]_{(x_f-2:x_f+3,y_f-2:y_f+3)}$ results in:

$$[1 \quad \Delta y \quad \Delta y^2 \quad \Delta y^3 \quad \Delta y^4 \quad \Delta y^5] * [QK] * [c]_{(x_f-2:x_f+3,y_f-2:y_f+3)}$$

This yields the interpolated B-spline coefficients which form a row of values (an example of this step is shown in the top-right picture in fig.15). The last portion of eq.44:

$$[QK]^T * \begin{bmatrix} 1 \\ \Delta x \\ \Delta x^2 \\ \Delta x^3 \\ \Delta x^4 \\ \Delta x^5 \end{bmatrix}$$

resamples the quintic B-spline kernel in a similar manner as in fig.14. When it right multiplies the rest of eq.44, it interpolates the gray-scale value we need from the interpolated row of B-spline coefficients obtained from the previous steps (an example of this step is shown in the bottom-left picture of fig.15). The overall process is pretty arduous, so a graphical explanation of the overall process is given in fig.15.

Lastly, examining the portion:

$$[QK] * [c]_{(x_f-2:x_f+3,y_f-2:y_f+3)} * [QK]^T$$

of eq.44, this term can be precomputed to increase the speed of the program [11]. This precomputation for biquintic B-spline interpolation requires a very large amount of storage (36 times the size of the padded B-spline coefficient array). But, the space required, in my experience, is worth the trade off for the speed improvement. As mentioned before, the largest computational bottleneck in the DIC analysis is the interpolation step when calculating the components of the hessian, so the reduction in computational time is worth the expensive memory requirement.

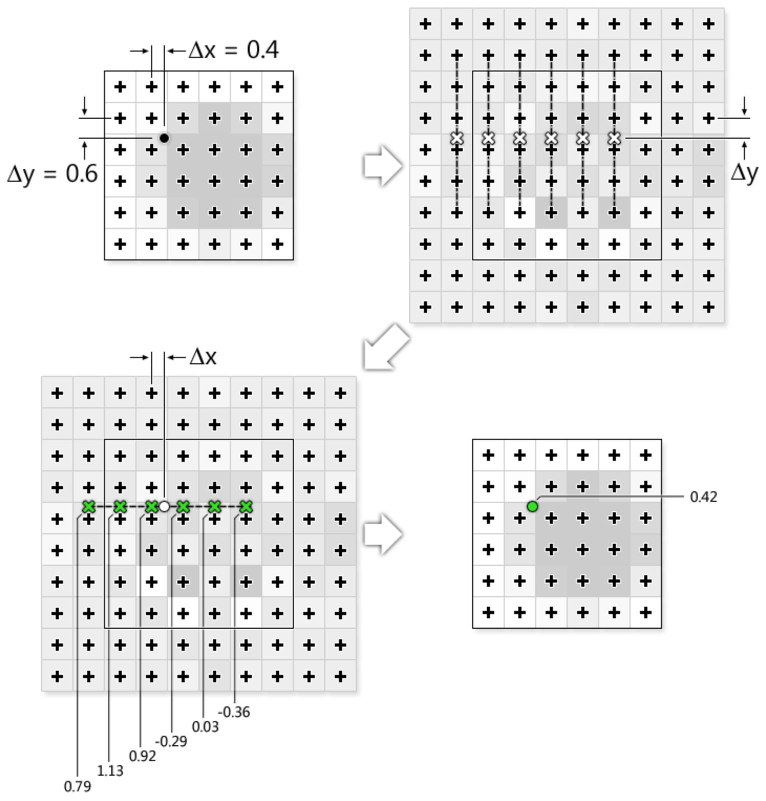


Fig.15 This figure is an extension of fig.13 and uses the same gray scale and B-spline coefficient values.The black crosses represent integer pixel locations. The black circle in the top-left is the subpixel point we're looking to interpolate. The steps involved are a graphical representation of eq.44.

At this point, the $g\left(\tilde{x}_{cur_i}, \tilde{y}_{cur_j}\right)$ quantity is calculable. The last quantities to address are $\frac{\partial}{\partial \tilde{x}_{ref_i}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right)$ and $\frac{\partial}{\partial \tilde{y}_{ref_j}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right)$. These quantities can be computed by taking the partial derivatives of an equation of the same form as eq.44 and setting Δx and Δy to zero (because these are integer pixel locations) to obtain:

$\frac{\partial}{\partial \tilde{x}_{ref_i}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * [QK] * [c]_{(x_f-2:x_f+3,\, y_f-2:y_f+3)} * [QK]^T * \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$	Eq.47
$\frac{\partial}{\partial \tilde{y}_{ref_j}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} * [QK] * [c]_{(x_f-2:x_f+3,\, y_f-2:y_f+3)} * [QK]^T * \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$	Eq.48

And just to reiterate, these quantities are precomputed for the entire reference image before beginning the IC-GN method. Furthermore, the b-spline coefficients used in eq.47 and eq.48 must be computed for the reference image as well.

At this point, all the quantities we need to compute the gradient and hessian for the IC-GN method are now known and calculable. This means we have the capability to calculate $\mathbf{p}^*_{rc}$ for an initial reference subset using eq.23, eq.26, and eq.27 with eq.11.

3.8 - Inverse compositional method summary

Because the explanation of the inverse compositional method is so long, I've decided to summarize the method in this section. The steps follow from [12] (but have been adapted for DIC) and are listed below:

Computation for entire DIC Analysis:

1. (*Optional*) Precompute $[QK] * [c]_{(x_f-2:x_f+3,y_f-2:y_f+3)} * [QK]^T$ for the entire current image.

2. Evaluate $\frac{\partial}{\partial \tilde{x}_{ref_i}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right)$ and $\frac{\partial}{\partial \tilde{y}_{ref_j}} f\left(\tilde{x}_{ref_i}, \tilde{y}_{ref_j}\right)$ for the entire reference image using eq.47 and eq.48.

Computation per subset:
3. Compute the "steepest descent images", eq.31-36.
4. Compute the GN-Hessian in eq. 26.
5. Set the initial $\mathbf{p}_{old}$ to the initial guess from NCC or neighboring deformation data

Computation per iteration per subset:

6. Compute the warped final current subset $g\left(\xi_{ref_c} + w\left(\Delta\xi_{ref}; \mathbf{p}_{old}\right)\right)$ using eq.44
7. Compute the gradient, $\nabla C_{LS}(\mathbf{0})$, using eq.23
8. Compute $\Delta\mathbf{p}$ using eq.11 with Cholesky decomposition
9. Update $\mathbf{p}_{old}$ using eq.27

10. Exit when the norm of $\Delta\mathbf{p}$ is small.

4 - Obtaining Full Field Displacements

As discussed in the previous section, we now have the capability to calculate displacement data for a single material point located at the center of an initial reference subset. The next step is to obtain a full array of displacement values. Usually, this is carried out by first selecting a region of interest (ROI) and then determining displacement data in a grid within the ROI. Then, the displacements are either reduced or interpolated to form a "continuous" displacement field. The best way, in my opinion, to carry this out is to use Bing Pan's reliability guided method [13]. An explanation of the method is given below, although the reader is directed to [13] for an in depth explanation by the author of this method.

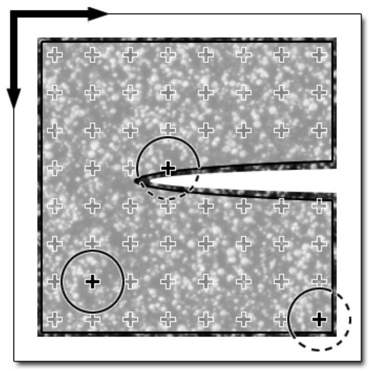


Fig.16 The figure above demonstrates a sample with an complex region of interest - in this case a specimen with a crack in it. The region of interest is the white highlighted area. Example subsets are shown superimposed on the ROI. The subset near the crack is a special case that demonstrates the need for subsets to be contiguous within the region of interest.

The RG-DIC method begins by selecting a seed point. This is the location of the center of the first reference subset analyzed in the DIC analysis. This point can either be selected in an automated fashion ([5], which uses the centroid of the region of interest; although this point may be poorly defined - i.e. if the specimen has a hole in the center) or manually. This subset is special because it is the only one that uses NCC, sift, or other global methods (as mentioned in the initial guess section) to obtain an initial guess. The rest of the subsets will use neighboring information as an initial guess as explained in the next paragraph.

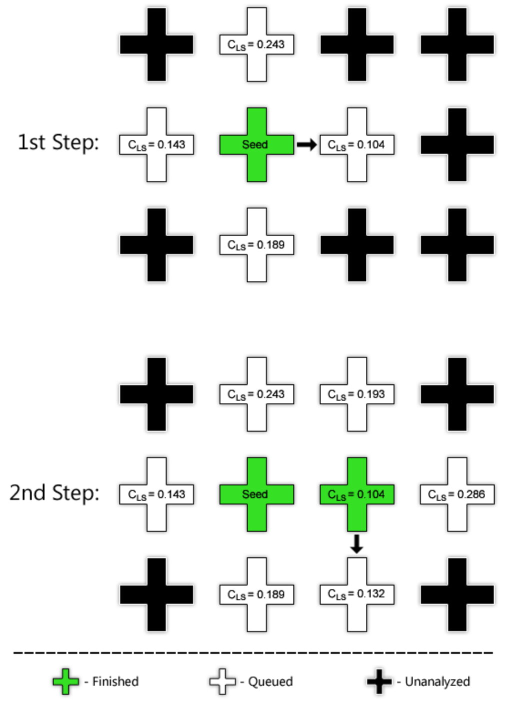


Fig.17 This figure graphically demonstrates the directions taken by the RG-DIC algorithm. The algorithm proceeds in the direction of the lowest C_{LS} value.

The above idea is implemented by selecting a seed point, calculating the corresponding deformation parameters and C_{LS} for a subset located at the center of this point, and then deactivating the point (done by forming a logical mask of the same size as the reference image initialized to false; analyzed points are set to true). A queue is formed (as a heap) and then C_{LS}, the six deformation parameters, and the location of the center of the subset, ($x_{ref\,c}$,$y_{ref\,c}$), are stored in the queue. The program enters a while loop and at each iteration, the top entry (this entry will have the lowest C_{LS} since the list is a heap) is copied and removed from the queue. The data for the loaded queue point is then added to the data plots. Next, the four surrounding material points, if they are still active or within the ROI, are analyzed using the displacement data from the loaded queue point as the initial guess for the nonlinear optimization scheme (note that you can also use the displacements gradients to improve the guess as well). The process is repeated until the queue is empty, at which point all contiguous points of interest will be analyzed. A flow diagram is included below and an example is given in fig.18:

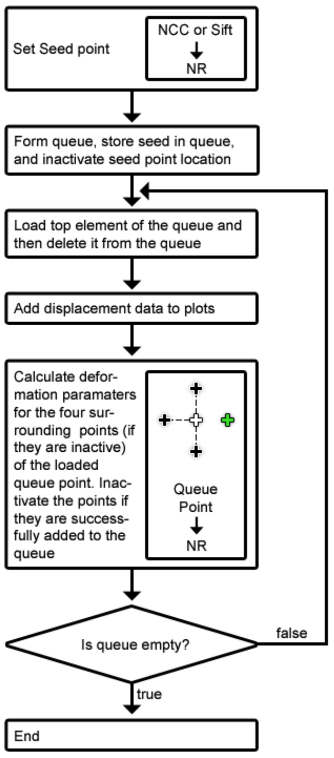

Set Seed point
NCC or Sift
NR
Form queue, store seed in queue, and inactivate seed point location
Load top element of the queue and then delete it from the queue
Add displacement data to plots
Calculate deformation paramaters for the four surrounding points (if they are inactive) of the loaded queue point. Inactivate the points if they are successfully added to the queue
Queue Point
NR
Is queue empty?
false
true
End

Fig.18 The figure above is a flow chart of the RG-DIC algorithm. In the neighbor calculation phase, the central white cross is the material point loaded from the queue. The three black crosses are unanalyzed points which are added to the queue. The green cross has already been analyzed and is skipped.

The benefits of this process are two-fold. First of all, it is robust in that bad data points (i.e. data points with high C_{LS}) are processed last, which prevents this data from being used as the initial guess for neighboring points. In addition, it is computationally efficient because only the seedpoint needs to use NCC or sift to obtain an initial guess. Moreover, the initial guess provided by the adjacent subset is typically quite good (as long as the subset spacing is relatively small) because displacement fields are, in general, relatively smooth.

The only problem I see with this method is that it must be done serially. One potential way to address this problem is to partition the region of interest and select multiple seed points that process in parallel. This idea was implemented in Ncorr.

5 - Obtaining Full Field Strains

Strains are more difficult to resolve than the displacement fields because strains involve differentiation, which is sensitive to noise. This means any noise in the displacement field will magnify errors in the strain field. One of the types of strain calculated in Ncorr is Green-Lagrangian strain, which is obtained by using the four displacement gradients as shown below:

$E_{xx} = \frac{1}{2}\left(2\frac{\partial u}{\partial x} + \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2\right)$	Eq.49
$E_{xy} = \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} + \frac{\partial u}{\partial x}\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\frac{\partial v}{\partial y}\right)$	Eq.50
$E_{yy} = \frac{1}{2}\left(2\frac{\partial v}{\partial y} + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2\right)$	Eq.51

The displacement gradients are actually directly obtained through the IC-GN scheme. But, these values are noisy, and thus must be "smoothed" in some way before calculating the strain fields. One possible way to do this would be to use a low pass filter to smooth the displacement gradient fields before using eq.49-51. But, the most common way to obtain displacement gradients, from what I've read, is to calculate them from the displacement fields [14], and basically ignore the displacement gradient information obtained from the IC-GN method. The strain window idea in [14] will be the main focus of calculating strains in this write up because it is the method used in Ncorr; it uses a least squares plane fit on a subset of displacement data (u and v) to find the plane parameters in eq.52-53. The equations for the planes are shown below:

$u_{plane}(x,y) = a_{u,plane} + \left(\frac{\partial u}{\partial x_{plane}}\right)x + \left(\frac{\partial u}{\partial y_{plane}}\right)y$	Eq.52
$v_{plane}(x,y) = a_{v,plane} + \left(\frac{\partial v}{\partial x_{plane}}\right)x + \left(\frac{\partial v}{\partial y_{plane}}\right)y$	Eq.53

The idea here is to form an over constrained system of equations of the form:

$\begin{bmatrix} 1 & x_{ref\,first\,i} - x_{ref\,c} & y_{ref\,first\,j} - y_{ref\,c} \\ \vdots & \vdots & \vdots \\ 1 & x_{ref\,last\,i} - x_{ref\,c} & y_{ref\,last\,j} - y_{ref\,c} \end{bmatrix} \begin{Bmatrix} a_{u,plane} \\ \left(\frac{\partial u}{\partial x_{plane}}\right) \\ \left(\frac{\partial u}{\partial y_{plane}}\right) \end{Bmatrix} = \begin{Bmatrix} u^*_{rc}\left(x_{ref\,first\,i}, y_{ref\,first\,j}\right) \\ \vdots \\ u^*_{rc}\left(x_{ref\,last\,i}, y_{ref\,last\,j}\right) \end{Bmatrix}$	Eq.54

The above equation uses the subset notation I've used previously for the subset window (see fig.1). Eq.54 can be implemented efficiently in the case of a rectangular ROI through [14], which converts the process into a convolution. Otherwise, it can be solved for using QR decomposition with column pivoting or through the process outlined in [15]. Once these parameters are solved for, they can be used in eq.49-51 to determine E_{xx}, E_{xy}, and E_{yy}. The overall plane fit idea is shown in the figure below:

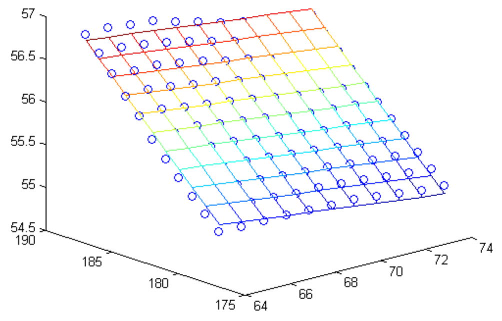


Fig.19 This figure is an example of a plane fit to a subset of displacement data points.

Eq.54 is solved for every material point processed in the DIC algorithm. The end result of this analysis is a full strain field. It is noted that the size of the strain window used in the strain calculation is variable and essentially acts as a smoothing operation; the larger the window, the larger the smoothing effect. Caution should be exercised when using this method to prevent the over smoothing of data.

6 - Concluding Remarks

At this point, the basic groundwork has been established for obtaining displacement and strain fields from images of samples undergoing deformation. Overall, this writeup was intended to explain the fundamentals of DIC with an emphasis on implementation.

7 - Citations

[1] H Lu and P D Cary, *"Deformation Measurements by Digital Image Correlation: Implementation of a Second-order Displacement Gradient."* Experimental Mechanics (2000).

[2] B Pan, H Xie and Z Wang, *"Equivalence of digital image correlation criteria for pattern matching."* Optical Society of America (2010).

[3] B Pan, K Li and W Tong, *"Fast, Robust and Accurate Digital Image Correlation Calculation Without Redundant Computations."* Experimental Mechanics (2013).

[4] J P Lewis, *"Fast Normalized Cross-Correlation."* Industrial Light & Magic (1995).

[5] B Pan, W Dafang, and X Yong, *"Incremental calculation for large deformation measurement using reliability-guided digital image correlation."* Optics and Lasers in Engineering (2011).

[6] S Baker and I Matthews, *"Equivalence and Efficiency of Image Alignment Algorithms."* Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition (2001).

[7] H W Schreier, J R Braasch and M A Sutton *"Systematic errors in digital image correlation caused by intensity interpolation."* Society of Photo-Optical Instrumentation Engineers (2000).

[8] P Thevenaz, T Blu and M Unser *"Image Interpolation and Resampling."* .

[9] M Unser *"Splines: a perfect fit for signal and image processing."* Signal Processing Magazine, IEEE (1999).

[10] R G Keys *"Cubic Convolution Interpolation for Digital Image Processing."* IEEE Transactions on Acoustics, Speech, and Signal Processing (1981).

[11] B Pan and K Li *"A fast digital image correlation method for deformation measurement."* Optics and Lasers in Engineering (2011).

[12] S Baker and I Matthews *"Lucas-Kanade 20 Years On: A Unifying Framework."* International Journal of Computer Vision (2004).

[13] B Pan *"Reliability-guided digital image correlation for image deformation measurement."* Optical Society of America (2009).

[14] B Pan, H Xie, Z Guo and T Hua *"Full-field strain measurement using a two-dimensional Savitzky-Golay digital differentiator in digital image correlation."* Optical Engineering (2007).

[15] D Eberly *"Least Squares Fitting of Data."* (2009).