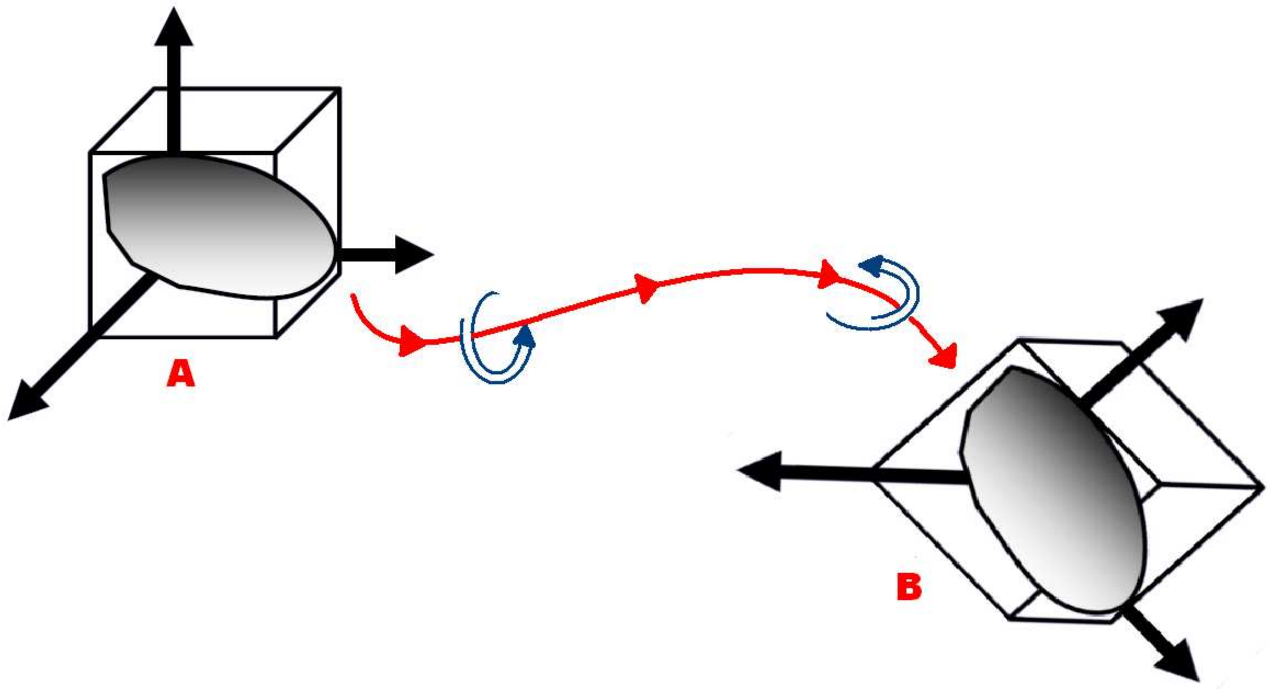
BTA/GE/09-20

Discrete 3D Digital Image Correlation (DIC) using particle reconnaissance



January 7, 2010

Rutger C.A. Smit

Department of Geotechnology

Delft University of Technology

Title	:	Discrete 3D Digital Image Correlation using particle reconnaissance
Author(s)	:	Rutger Cornelis Aart Smit
Date	:	January, 7th 2010
Professor(s)	:	Dr. Ir. Dominique Ngan-Tillard
Supervisor(s)	:	Dr. Ir. Dominique Ngan-Tillard
TA Report number	:	BTA/GE/09-20
Postal Address	:	Section for GeoEngineering Department of Applied Earth Sciences Delft University of Technology P.O. Box 5028 The Netherlands
Telephone	:	(31) 15 2781328 (secretary)
Telefax	:	(31) 15 2781189

Table of contents

1 Introduction

This bachelor project is done for the study of strain localisation in geomaterials. The conventional material testing only describes the fabric behaviour globally. In order to create better understanding of how sand/rock particles move under compression, micro X-ray CT-scans are made of specimens from the material in question. An X-ray CT-scan alone however, is not enough to properly analyse the fabric movement. For this purpose, Digital Image Correlation (DIC) is applied to obtain 3D images of the fabric movement.

Well known is the classic continuum 3D DIC. This bachelor project will however make use of discrete 3D DIC. One advantage of this method compared to the classic continuum DIC is the ability to describe the individual particle movement.
Previous research on this field has shown that there are difficulties with the identification of individual particles. For this research it was attempted to develop a new and easier way to detect the grains with the help of the programs Avizo 6, Matlab and Excel.

The ultimate goal was to successfully apply discrete 3D Digital Image Correlation using particle recognisance. This report will therefore elaborate the way that the analysis was performed, describe the instruments and tools that were used and discuss the results obtained with this method.

2. Research outline

This chapter will focus on the materials and the equipment that was used for the purpose of this research. Then in chapter 2.4 the roadmap that was used to retrieve the obtained results will be discussed.

2.1 Sand type and grain size

In order to get to know the program Avizo and its modules, .DECOM files were used of sand that were constructed for other purposes. With the use of these files the different grain sizes and their (dis-) advantages were analyzed and described.

For this research the first goal was to attempt to identify the grains in the image before grain-movement and to find the same grains again in the image after movement. Therefore it was important to develop a solid way of detecting grains but also to use the right sand type and size. The sand type known as "Maas sand" was used for all scans that were made in this BSc project. Although this sand type contains mainly quartz, there are also other minerals present. From the .DECOM-files of Maas-sand that were created for previous experimental testing of the mirco CT-scanner, it was found that the grey level differences between the quartz and the non-quartz minerals were close enough to still properly identify individual grains.

For a proper analysis of a 3D image, it is required to have the right grain size distribution for the test sample. It is hard to identify a grain when it consist of too few voxels. If the grain size is too big however, there will be too few grains to properly analyse the sand fabric movement. For example, if there are only 20 or less grains in a measurement volume, it is difficult to observe trends in grain movement.
In order to obtain the right grain size, 'old' .DECOM-files of images from Maas sand with three different grain sizes were analysed. The scans were made of loosely packed Maas sand of the grains from the same configuration in layers in the same container. Each section, with a specific grain size, was analysed the same way:

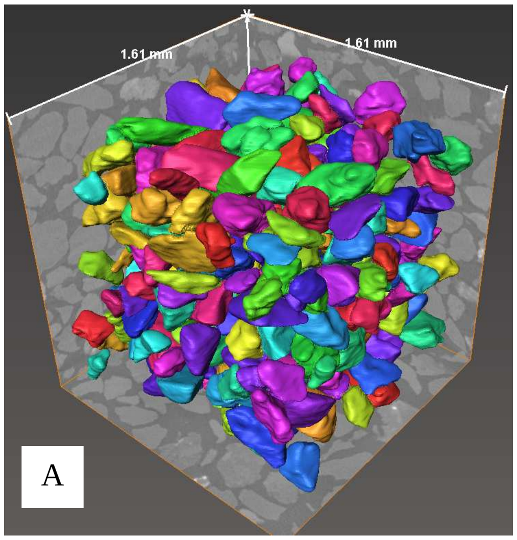
- A section of 268 DECOM files from the given grain size is selected (voxel size is 0.006 mm)
- The image is cropped in such a way that a cubic volume with ridges of exactly the same size (1.61 mm x 1.61 mm x 1.61 mm) is obtained.
- Image analysis. (See for script in Appendix 1)
- Grains that intersect the volume boundaries are not shown or analyzed

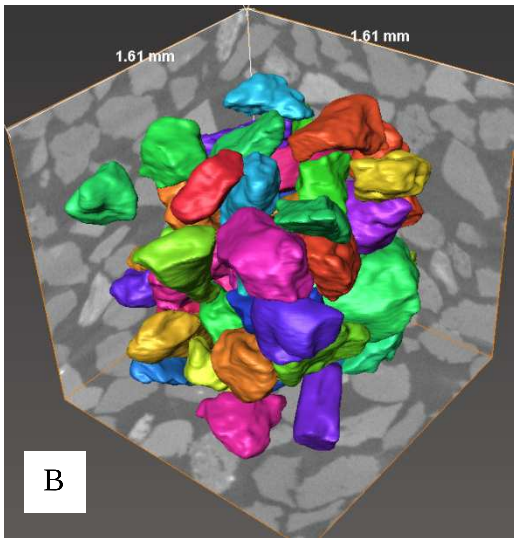
The results obtained are displayed in figure 1 below.

It turned out that it is quite hard to apply segmentation on the smallest grain size with a diameter of 0.063 mm – 0.106 mm. When we look at the 3D image of the grains more closely one can actually see that some separate grains are actually still glued together. It would take a long time to separate all grains, especially since there are already so many grains in a small area like this. Another downside to this grain size, is that the grains consist of very few voxels and therefore the shape parameters of grains are very dependent on the threshold that is applied. For these reasons this grain size was not used for further the development of the discrete DIC method.

The medium grains have a diameter ranging from 0.106 mm – 0.210 mm. In the measuring volume there are now 300 grains, compared to more than 2000 particles in the same volume for the smaller grains. Since the grains consist of much more voxels the shape parameters are much better defined and the detail of the grains is higher. Observing the 3D image of this particular section more carefully, it seems that there haven't been too many grains that have erroneously been 'glued'.

The large grains with a diameter ranging from 0.210 mm – 0.300 mm are even easier to distinguish than the medium grains. One downfall of these grains might be that there are to few in the control volume to properly analyse transformation of grains.





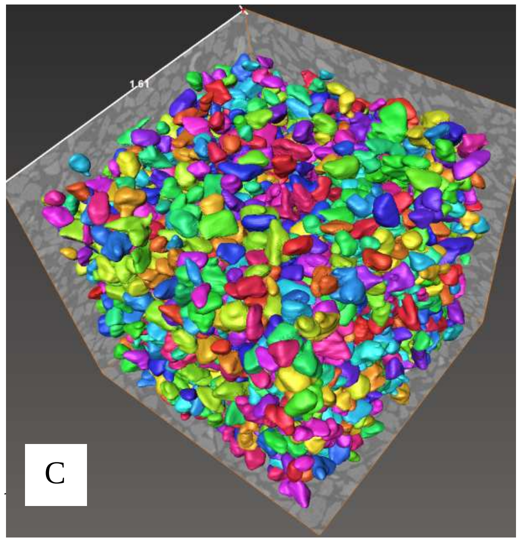


Figure 1:

Three 3D images of grains with different sizes in a cubic control volume with edges of 1.61 mm (268 voxels).

A: medium grains (0.106mm)
B: large grains (0.210mm)
C: grains with the smallest diameter (0.063mm)

For the development of the processing method sand broken by a micro-Deval was used. This was done to increase the range on the shape parameters from the grains (such as Volume and Area3D). This way it was easier to recognize and detect the grains from the binary images created with Avizo.
In the first test that was performed the largest grain size was chosen to work with because these grains are easy to detect and thus easy to correlate. For the final test the medium grain size was used since these are also quite easy to distinguish but their movement shows a better way of how the grain matrix is reshaped as a compression or rotation test is performed.

2.2 Research mounting

Two tests were performed to develop the image processing methods and the data processing methods necessary for the Digital Image Correlation (DIC). The first test was light compression of unconsolidated sand samples and the second movement of unconsolidated sand by the rotation of a wooden cylinder. From now on these tests will respectively be referred to as the 'compression test' and the 'rotation test'.

2.2.1 The compression test

In order to take a proper image, the casing of the research mounting had to be very transparent for X-rays because for obvious reasons it is not desirable that the casing disturbs the image of the grains. In or on the casing a reference axis needs to be created which is also transparent. The second property the research mounting had to be taken into account was that it needed to fit in the holder of the CT-scanner and. As the mounting is turned by the holder the imaged part of the research mounting needs to remain in the same position, the mounting therefore needs to be straight. Finally it is also required that the grains can be 'settled' or compressed in the tube that they are in.

The first research mounting that was used, consisted of a 1 ml injection tube (inner diameter: 4.9 mm) with needle casing. In order to let the CT-scanner create a good image, it needs to be as close to the research sample as possible. Therefore the top of the injection tube needed to be sawed off in order to remove the broad protuberances that would have been in the way of the X-ray emitter (Figure 2).

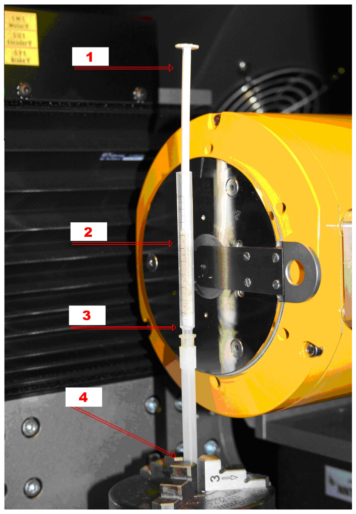


Figure 2:
Research mounting of the 'compression test'.

1. *Plastic cylinder with which pressure can be induced on the sand*
2. *Plastic 1 ml tube with the sand*
3. *At the bottom of the tube a piece of filtering paper and the connection with the needle casing.*
4. *The bottom of the needle casing is clamped by the standard of the CT-scanner.*

It is not desirable that the mounting is moved or displaced in between the two imaging processes (before and after compression). A reference axis needs to be created on the casing to, if necessary, digitally correct any unintended transformation. For this test, the tube of the mounting was perforated on several places as a reference.

In order to minimize the displacement of the research mounting during the compression, the mounting was not removed from the holder. Light compression was applied on the grains with a plastic cylinder.

The imaging of the grains in their compressed state was more of a problem since it is observed that the plastic tube was displaced a bit by the compression. The result was good enough to be analyzed, but there was some displacement which needed to be corrected. Another problem was that the perforations, which were made as a reference, were not clear enough on the images to use them.

2.2.2 The rotation test

The difference between the compression test and the rotation test is that instead of compressing the sample with a plastic cylinder; a wooden stick with a rough surface was fixed in the middle of the tube and rotated to force translation and rotation of the grains.

For this second test, also a few other features were changed. To prevent bending of the mounting, the top and bottom of the injection tube were cut off and the remainder of the tube was glued on a glass cylinder to make the mounting more rigid and straight. Also a different type of reference was used. Grains of sand were glued on the outside of the tube with tape as markers (Figure 3). This time no obvious displacement of the mounting was observed which proves that the second mounting was much sturdier.

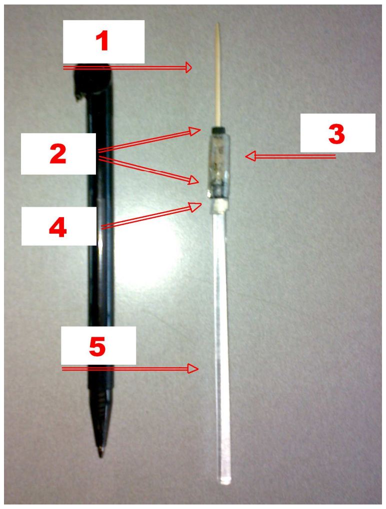


Figure 3:
Research mounting of the 'rotation test'.

1) *the wooden tooth stick*
2) *two rubber cylinders to fix the tooth stick in the middle of the tube*
3) *Cut-off plastic1ml tube with the sand and tape with marker grains around it as reference.*
4) *The bottom of the tube is glued to the glass cylinder.*
5) *Glass cylinder*

A third type of reference was tested for the use of this project. For the first rotation tests, a thin copper wire with a diameter of 0.18 mm and an even thinner iron wire with a diameter of 0.05 mm were attached around the tube with plastic tape to create a reference axis. The first images of these scans however, quickly proved that both the copper wire as well as the iron wire gave so much radiation on the images that segmentation would be impossible. In figure 4 and 5 the results from imaging with respectively grains and iron wires as reference.

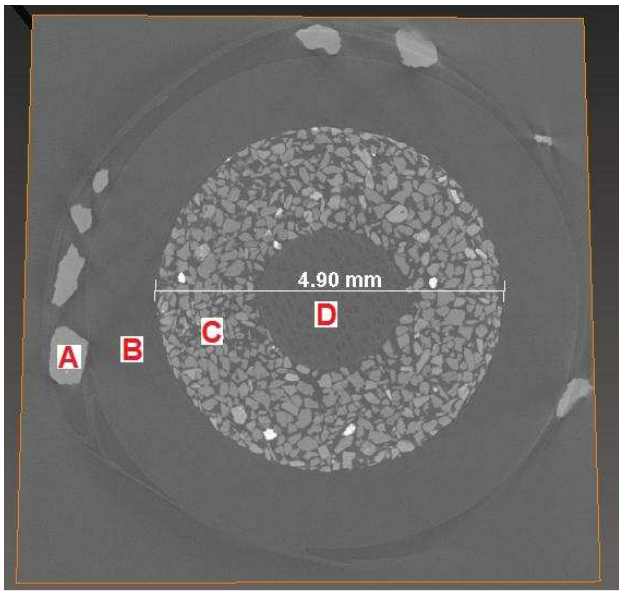


Figure 4:
A: reference grains
B: plastic cylinder
C: grain matrix
D: wooden rotation stick

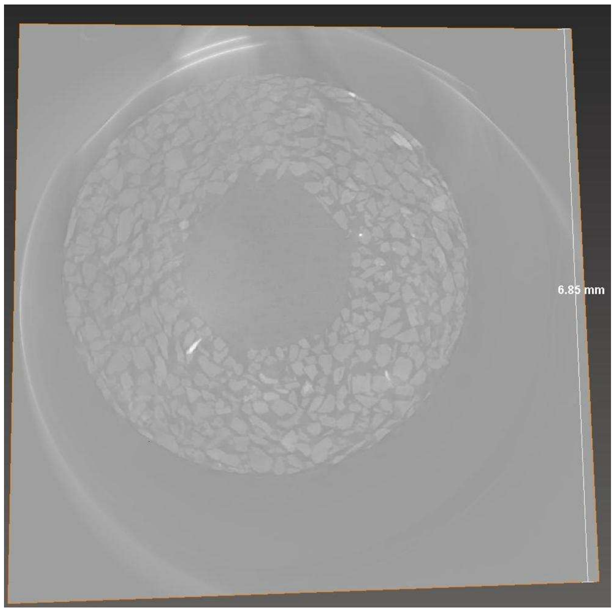


Figure 5:
This image shows the effect of using an iron wire as reference. The image cannot be segmented because by the radiation of the wires.

2.3 The micro X-ray CT-scanner

The CT-scanner is a widely used tool and showed to be very useful in the field of micro-rock mechanics. For the creation of each 3D image the scanner takes 7 images, each image sampled from 720 different angles. After the scanning process the two worst images are removed with Qwin. By aligning the other images, the final image is obtained.
The images that were obtained from the scan have a voxel size of 3 μ. After re-sampling however, the resolution decreases to 6 μ. Thereafter they have to be transformed to DICOM files before they can be used in Avizo.

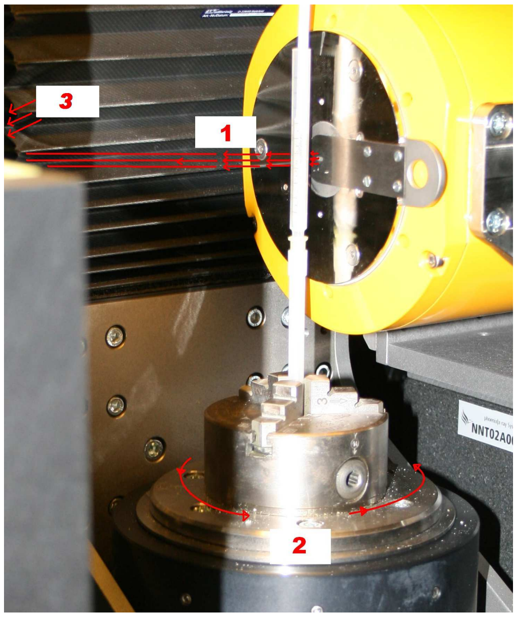


Figure 6: CT-scanner with sample

1) *X-ray are emitted from the CT-scanner through the mounting*
2) *The holder turns the mounting with 0.5 degrees at a time during the imaging process.*
3) *The camera records the X-ray that have passed through the sample and have been alternated.*

2.4 Process method walkthrough

After the Qwin files are converted to DECOM files that can be read by Avizo 6. A series of steps have been performed to obtain images of the translation and rotation of grains. The roadmap that was used to process the two tests that were performed for this BSc project, is shown in table 1.

Program	***Process***	***Step***
Avizo	Image Processing	*Alignment Check*
		Cropping of images
		(Multiple) Thresholding
		Hole filling
		Grain separation
		Removal erroneous separations
		Labelling
		Obtaining grain shape parameters
Excel and Matlab	Data Processing	*Threshold check*
		Sorting grains
		Filtering grains
		Comparing grains
		Obtaining the Transform Matrix
Avizo	Results	*Selecting grains*
		Coloring grains
		3D imaging

Table 1: *Global overview of the roadmap used for processing the compression test and the rotation test.*

3. Image processing

The input images derived from the CT-scanner are grayscale images, which can be seen as a collection of voxels (3d pixels) in which properties of grains cannot yet be identified. For this BSc thesis the program Avizo Fire 6.0 was used to process the images. The scripts that were made can be found in Appendix 1.

The goal of image processing is to create a 3D image in which all grains can be separately addressed and where shape parameters, such as the volume and surface area of a grain, can be obtained. In order to be able to identify separate grains, the image has to undergo a couple of processes which will described in detail in this Chapter.

3.1 Alignment of the sample images

As discussed in the previous chapter, two tests have been performed for the development of the discrete DIC method: the 'compression test' and the 'rotation test'.

In the compression test big grains (diameter 0.210 mm to 0.300 mm) were used and the amount of vertical displacement of the grains was relatively small (<20 voxels). In figure 7 a 2D image before and a 2D image after compression are displayed. To correct for the error that emerged from the bending of the plastic tube, manual translation has been applied to align the principal axis of the two images. The correction that needed to be applied was 0.168 mm in the x-direction and 0.024 mm in the y-direction.
This transformation was however not applied on the image. The reason not to apply the translation (and rotation if present) is the desire not to digitally deform the grains in this process. Precautionary these values were be added to the values obtained from the actual movement of grains from the compression.
Of course transformation before the segmentation should lead to the same results but this is not tested in this bachelor project.

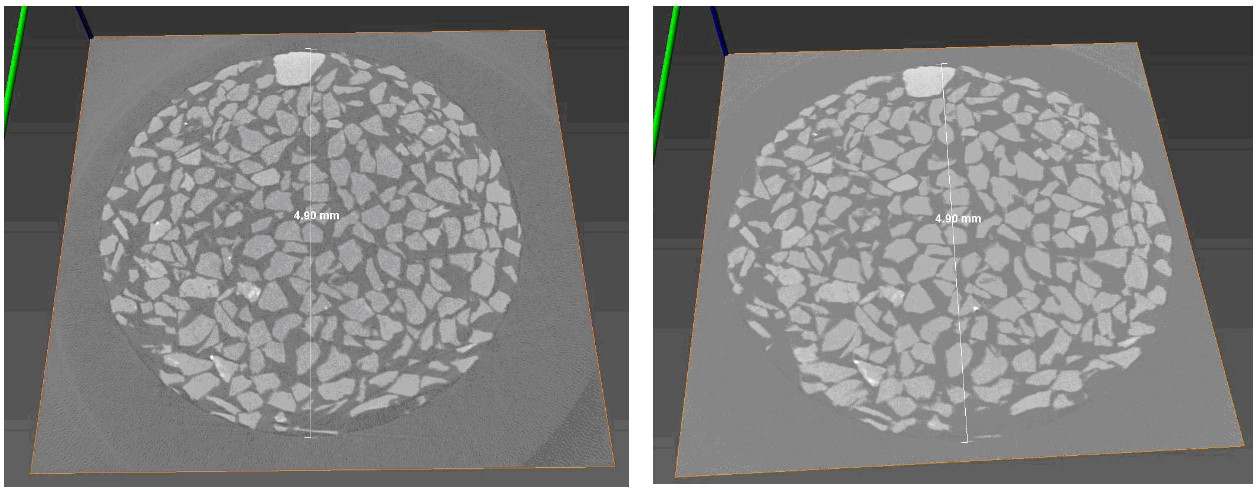


Figure 7: *Images obtained from the first CT-scan. Right: before compression. Left: after compression and 20 OrthoSlices lower.*

In the rotation test the grains with a diameter ranging from 0.106 mm to 0.210 mm were used. (See figure 9). Even though no visible displacement of the research mounting had taken place between the two images the feature 'align surfaces' was applied on a couple of reference grains, to check whether translation or rotation needed to be applied on the entire volume.

As can be seen in table 2, the translation suggested by Avizo is smaller than 1 voxel in each direction and the rotation smaller than 1/20 of a degree in each plane. This displacement was neglected since it is very small compared to the kind of displacement that we are looking for with in this test. (>10 voxels)

Translation:

dX	dY	dZ
-0,00423	-0,00186	0,001119

Rotation:

Angle:	0,039565	[degrees]
Around X-axis	**Around Y-axis**	**Around Z-axis**
0,3461	-0,06428	-0,93596

Table 2: *The transformation required to align the surfaces of the reference grains*

Another way of measuring the amount of undesired transformation in Avizo is landmark registration. This feature however turned out to be less precise because of human errors and more time consuming than align surfaces.

3.2 Binary image creation

Before a 3D image of a grain can be created, a reliable binary image in which separate grains can be defined needs to be developed. The quantification module in Avizo proved to contain powerful segmentation tools useful for the purpose of this research.

3.2.1 Thresholding

The first step in the segmentation process is the selection of a range of greylevels for which only the grains are selected and the, somewhat darker, pore spaces are left open. This is called "thresholding".

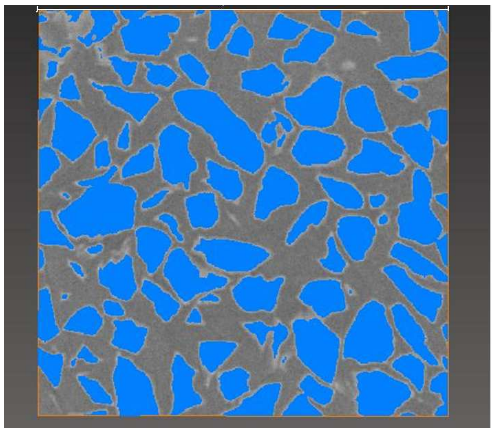


Figure 8:
'Easy' thresholding on a relatively small volume (the borders are 2.7mm each)

It would be really useful if a grain of a certain composition would always be displayed at the same greylevel intensity. Unfortunately this is not the case and the thresholds that needed to be applied on the images were never the same. During this bachelor project it remained unclear why the greylevel range and intensity vary with each CT-scan taken, even it is the same sample that is imaged.

Another difficulty with thresholding arises from the following: As an image is made with the CT-scanner, the intensity of the X-rays decreases as they penetrate the outer layers of the grains. Therefore the inner grains of a mounting have a different greylevel intensity than the ones on the outside. This effect can partially be undone by applying 'beam hardening' in Qwin but this effect could not be totally undone.

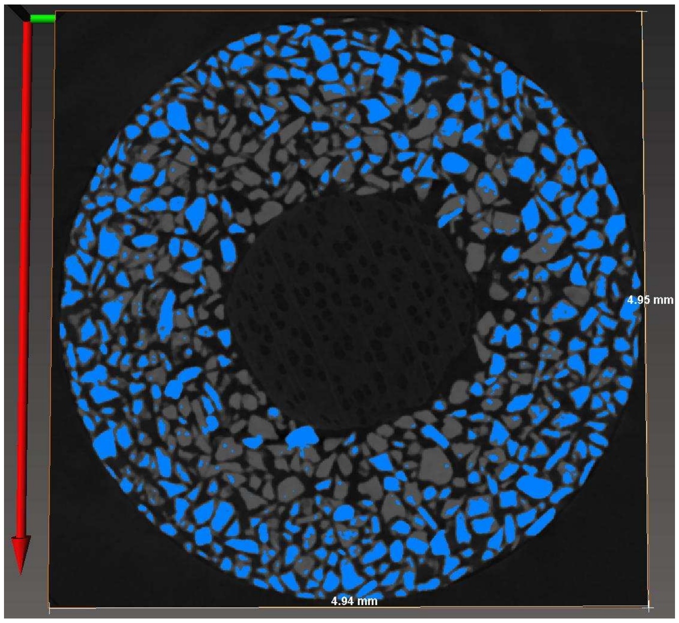


Figure 9:
The binary image (blue) is placed on top of the grayscale image. This illustrates the difficulties with thresholding. While grains in the outer rings are already covered by the threshold, the inner grains are not.

For the compression test, a somewhat smaller volume was analyzed. Therefore only one threshold needed to be applied on the entire volume. (Figure 8) The rotation test is however analyzed over the entire cross section of the cylinder. In this case the effect of diminishing X-ray intensity is affecting the way thresholding needs to be applied (Figure 9).

To bypass this problem, multiple thresholds on multiple shells need to be applied. As these shells are closer to the center, the range of greylevel it covers needs to increase.
To do this a series of hollow cylinders are cut from a voltex (primitive 3D image from the grayscale image) with the help of the volume edit feature. On these separate shells the different thresholds are applied. For example: on the image of the rotation test, where the grain have already been stirred, the cylindrical tube was devided in four shells with the following lower thresholds from outer to inner ring: 8800, 8735, 8685, 8600. All upper thresholds were set at 20000: the color white.

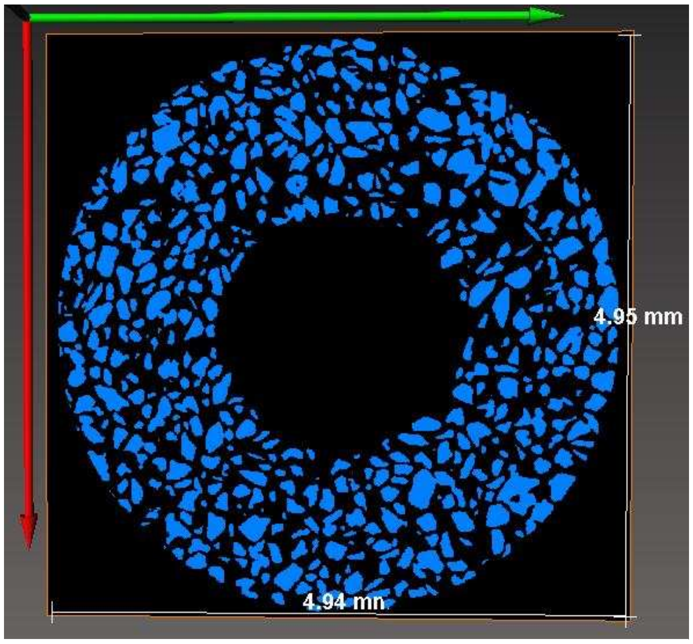


Figure 10: *An improved binary image consisting of 4 threshold shells*

The method described above is applied on both images that were obtained from the rotation tests but not written in script. Figure 9 displays the segmentation before the multiple thresholds were applied and Figure 10 displays the binary image with adjusted thresholds.

After an acceptable threshold has been found and applied, the feature 'hole-filling' is applied. This is a function which can fill small intrusions of the grains without also filling actual pores in grains. The goal of this feature is to remove small contaminations and noise created with the CT-scan.

3.2.2 Separation of grains

At this point, most of the grains appear to be 'glued' to another grain while in fact these connections are of course the grain-grain contacts. To separate the grains on their contacts, both the binseparate command from the Quantification module and the watershed – algorithm can be applied (the latter is used in the scripts, which can be found in Appendix 1). The results of both methods were quite similar. It is however not possible to perform manual correction on the separations with the binseparate command while this actually is possible with the watershed algorithm. Therefore this algorithm was used for all images that needed to be analyzed for the purpose of this project.

Even though the watershed-algorithm is a powerful tool to separate grains, it is not perfect. Erroneous grain-grain contacts are detected by this algorithm, which results in partition of solid grains. In order to these erase erroneous separations, an analysis is performed on the separation planes. It turns out that mainly separations with a very large surface area tend to be erroneous separations. To prevent removal of valid separations, these separations need to be inspected manually.

Five kinds of separations can be defined:

1) ***The separation is completely valid (a true grain-grain contact)***

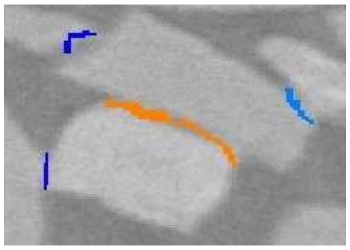


Figure 11:

The orange separation is large in area but it is, in fact, a valid separation.

In this case the separation is correct and therefore not removed.

2) ***The separation is mainly valid:*** the grains are not separated at the correct position but the separated grains are only slightly concave/convex and therefore still useful for analysis.

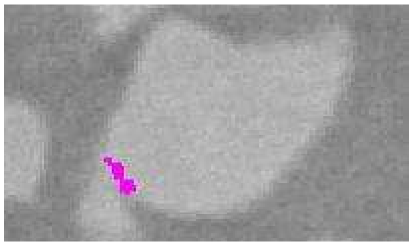


Figure 12:
The purple separation plane separates two grains but not entirely in the right spot. In 3D the grain seems bigger (and more convex) than it actually is.

In this case the separation is not entirely correct but if the error is not too big, the separation is left intact. The alternative (when the error is considered to be too big) both grains are left out of the analyzing process.

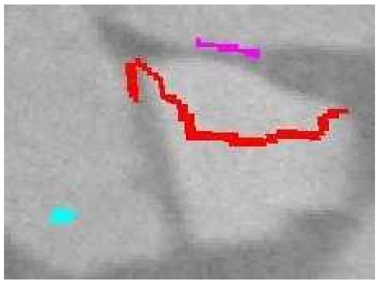


Figure 13:
The red separation mainly erroneous since it divides the grain on the left in two parts. But it also separates this grain from the grain on the right.

3) ***The separation is semi-valid:*** one part of the separation is valid and actually separates grains on their grain-grain contacts but another part of the separation creates untrue splits in solid grains.

This type of separation can be corrected in the following way:

a) Copy the original image with all separations
b) Filter this image so that only the separation in question is left
c) Crop this image in a way that only the valid part of this separation is left.
d) Crop the image again to resize it to its old scale.
e) Use a logical_or command to add this cropped separation to the other separations that need to be filtered out.

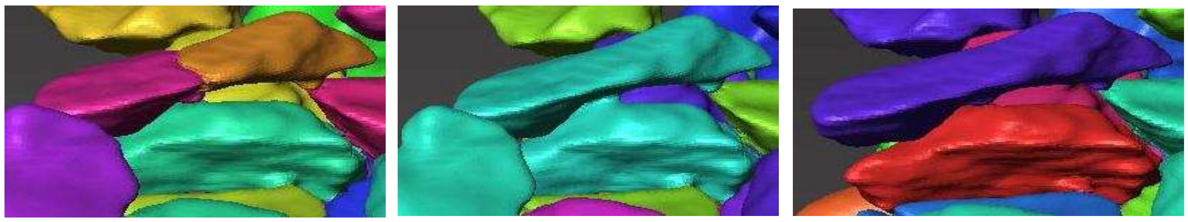


Figure 14: *Left: the separation separates two grains on the grain-grain contact but also divides the upper grain in two halves. Middle: When the separation is removed, the upper grain is no longer divided but the upper and lower grain remain merged on their grain-grain contact. Right: The two grains have successfully been separated*

4) The separation is either valid or false but the grains that are to be separated fall partially out of the measuring box and are therefore removed anyway.

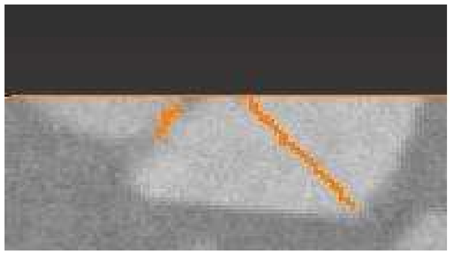


Figure 15:
The orange separation plane both creates a valid separation and an invalid separation on grains that intersect with the measuring frame.

5) The separation is completely erroneous.

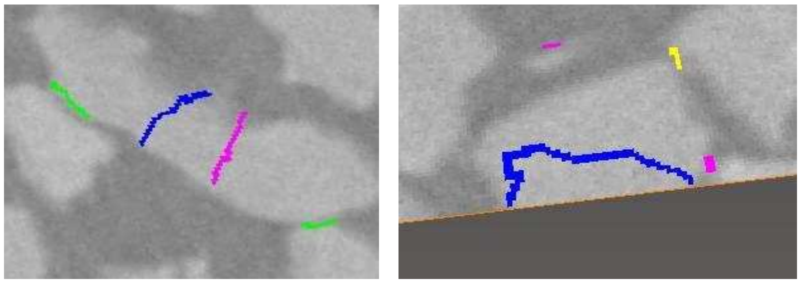


Figure 16:
Left: Two separations that divide one solid grain in 3 pieces. Right: the blue separation divides a grain that intersects with the measuring frame.

The separations need to be removed in order to create correct

In Avizo it is not possible to select something based on its index number instead a filter on Area3D or Volume3D can be applied. For example: If an erroneous separation with an Area3D of X mm^2 needs to be selected, the filter has the following form:

X-0.0000001 < Area3D < X+0.0000001

When all erroneous separations are selected, the logical_sub command can be used to subtract these separations from the watershed image. With this corrected watershed image the corrected separated grain image can be obtained.

Another error, which occurs with both the binseparate command and the watershed algorithm, is that some grains remain glued together on their grain-grain contacts. No solution has been found for this problem but these most of these 'glued-grains' are removed with a filter on the Shape_VA3D during the data processing. This topic will be discussed more thoroughly in chapter 3.3.

3.3 Grain identification and filtering

Now that a binary image is obtained where the grains are separated, we need to identify the grains (i.e. address a unique number to each grain). This is done with the labeling function form the Quantification module. Before we can properly analyze all grains, we first need to remove all residual noise. Grains that are very small and have the size of only a few voxels are very likely to be nonexistent, wrongly segmented grains and need to be removed. For both the compression test as well as the rotation test a value of 0.001 mm^3 was chosen as a minimal threshold for the 3D volume of the grains in the control volume. This value was chosen with the help of the histogram viewer in the result viewer of Avizo.
Another filter is applied on the boundary box of the grains to remove all grains that are in contact with the borders of the control volume.

The last filter that is applied in Avizo, is a restriction on the Shape_VA3d of a grain. The definition of this feature is:

$$Shape\_VA3d = \frac{(Area3D)^3}{(36 * \pi * Area^2)}$$

With:

$Area3D$ = The total surface area of the object

$Area$ = The internal cross sectional area of the object

The grains were filtered with a Shape_VA3d higher than 4 for both the compression and the rotation test. As it turns out, the larger the Shape_VA3d value is for a particular grain, the more convex the grain is (see Figure 17). This group of highly convex grains consists of 3 main groups:

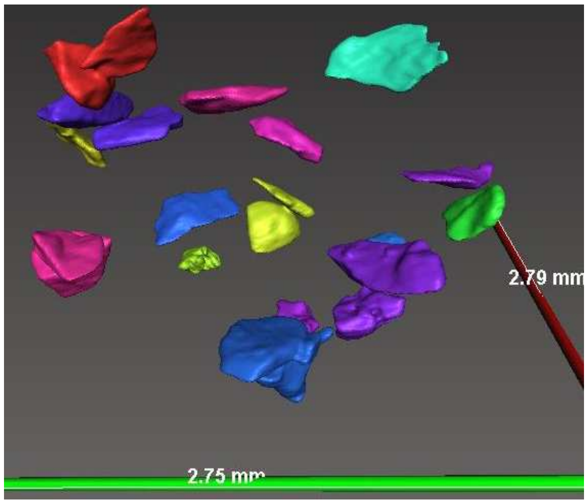


Figure 17: *Grains Shape_VA3d parameters bigger than 4*

1) Unsegmented grains, where two or more grains are glued.
2) Grains that contain minerals that have low graylevels. The threshold that was applied simply did not cover some parts of these grains which makes these grains appear to be convex.
3) The last group of grains is just convex of nature caused by intrusions, pores inside of the grains. Grains being really flat and thin in one direction also belong to this group.

The contrary is also true: the smaller the Shape_VA3d value of a grain, the rounder it appears to be (see Figure 18). No filtering based on low Shape_VA3d values has been done in Avizo. This given might however be used in the future for grain filtering. Why this is will be explained in chapter 4.1.

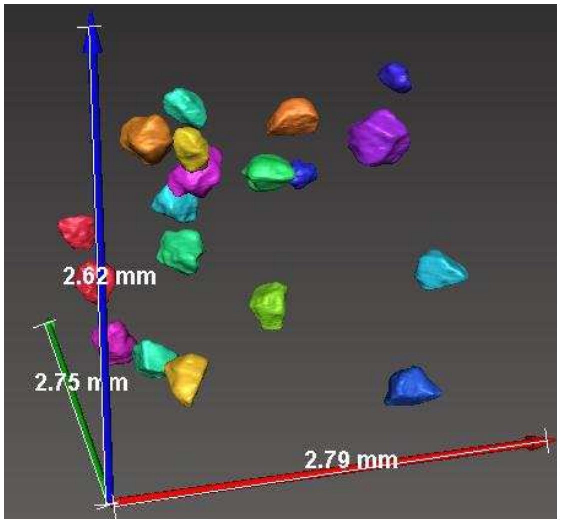


Figure 18: *Grains Shape_VA3d parameters smaller than 1.5*

3.4 Analysis

Now that all noise and cut-off grains are removed, the binary image is ready to be analyzed and shape parameters of grains can be obtained with the I_analyze feature. A question one might ask here is which shape parameters of a grain can be helpful for the identification of a grain. In the two tests that were performed for the purpose of this bachelor project Area3D and Volume3D have been chosen to be the main criteria for finding corresponding grains. Other parameters that need to be obtained are the BaryCenters of the X,Z and Z direction and all second order moments of inertia of each grain. These parameters are required to eventually calculate the transformation matrix for each grain from its old to its new state.

In the Avizo result viewer it is not possible to perform any computations or comparisons between numbers and can only be used to display the shape parameters of grains. It is however necessary to find corresponding grains based on their shape parameters in order to perform DIC. All data processing therefore needs to be done with the help of other programs. Both Excel and Matlab have been used to compare the values obtained from Avizo; perform computations and create filters. How this is done, is discussed in Chapter 4.

4 Data processing

This chapter will elaborate on the way an attempt was made to find grains based on their shape parameters. Secondly it will describe how the some of the shape parameters are used to obtain the transformation matrix, which describes how particles have moved with the compression of rotation test.

4.1 Identification and comparing procedure of grains

After the shape parameters of the grains are copied from the result viewer of Avizo to Excel, the grains are sorted by their Area3D. Now a value is assigned to each grain which represents its ranking according to its surface area. (1 = largest surface Area, 2 = second largest, etc.) This value was called the 'Excel index' of the grain.

4.1.1 Grain reconnaissance for the compression test:

The first test performed for this BSc thesis, was the compression test. Before a Matlab script was written, some manual comparisons between grains from before and after compression were performed of which was certain they were one and the same. From these comparisons, it was derived that their difference in Volume3D was never bigger than 0.0005 mm^3. It could also be observed that their horizontal movement was never more than 0.036 mm (6 voxels). The last noticeable item for the grains that have been compared, is that their Excel index never differed more than 10.

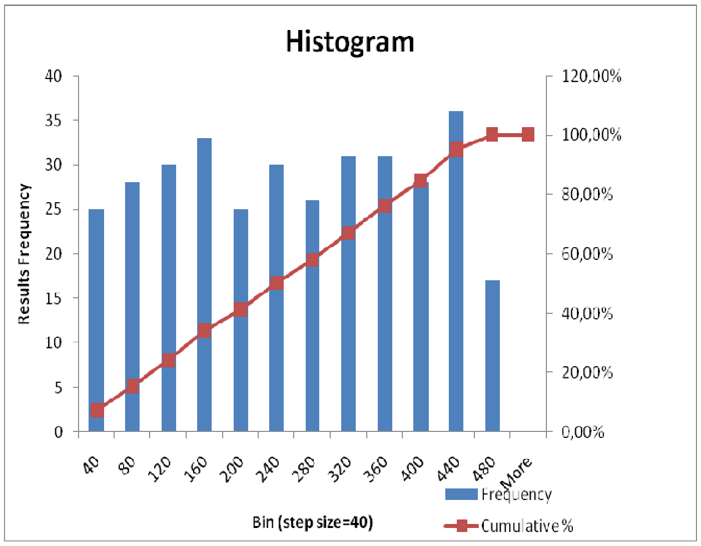


Figure 19: *This histogram shows the amount of results obtained with compare.m on the data of the compression test per 40 grains.*

With the use of these observations a Matlab script was created called "comparegrains.m" (see Appendix 2). In this script grains are compared based on the criteria that were found from manual comparison of the grains.

4.1.2 Grain reconnaissance for the rotation test:

The initial results obtained for the rotation were not as expected. When a second attempt was made to obtain results, less data was used and some additions and major improvements on the script in Matlab were applied. This sub-chapter will therefore first discuss the initial adjustments made on the Matlab script with regard to the compression test. Secondly this sub-chapter will discuss why the initial script was unable to properly process the data and why the improved script is.

The script for the identification and comparing procedures of the grains from the rotation test differs from the script written for that of the compression test. This has three causes:

1) The tests were different. In the compression test, only a small vertical displacement could be observed. The lateral displacement remained small. Therefore it was possible to apply a filter on the overall displacement of a grain. I.e. if a grain from before the compression was compared with a grain from after the compression and the displacement had appeared to be more than a certain amount, it was possible to reject the comparison based on this given.
   The same filtering method could however not be applied with the same strictness to the rotation test, because there is a lot more grain movement in this test.
2) Since smaller grains were used for the rotation test, the shape parameters of the grains were smaller and tended to be closer together than was the case for the grains of the compression test. To illustrate the difference: with the compression test 465 grains in the research volume were found whereas with the rotation test almost 2000 grains were found in a slightly smaller volume .
   Because of these differences, it was not possible to compare the grains from the rotation test in the same way the grains from the compression test were compared. Instead of a fixed amount of difference in Vol3D as a filter, now an increasingly smaller difference in Vol3D was required to be the filtering criteria in order to find the correct corresponding grains.
3) The way grains were compared in the compression test was limited. Instead of only comparing grains that had a certain range of Excel indices and agreeing with every result that was found, a new procedure was developed where all grains could be compared and only the best result was chosen.

The initial data processing method for the rotation test:

The selection with a filter based on the lateral and vertical displacement was also done for the rotation test. But for the reasons mentioned above, this could not be done too strictly. On the vertical displacement the results for comparisons larger than 0.7 mm were rejected and also the comparisons of grains that had a larger cumulative displacement than 1.5 mm were rejected.

That this filter, on the amount of displacement, turned out to be a mistake will be shown in Chapter 5.2. The reasons to apply a filter on the lateral displacement of grains were: To reduce the amount of erroneous grain comparisons; reduce the total amount of results (as this speeds the process up a lot) and to show that the smallest displacements, are clearly on the outside of the volume while the bigger displacements tend to be found closer towards the centre of the research mounting.

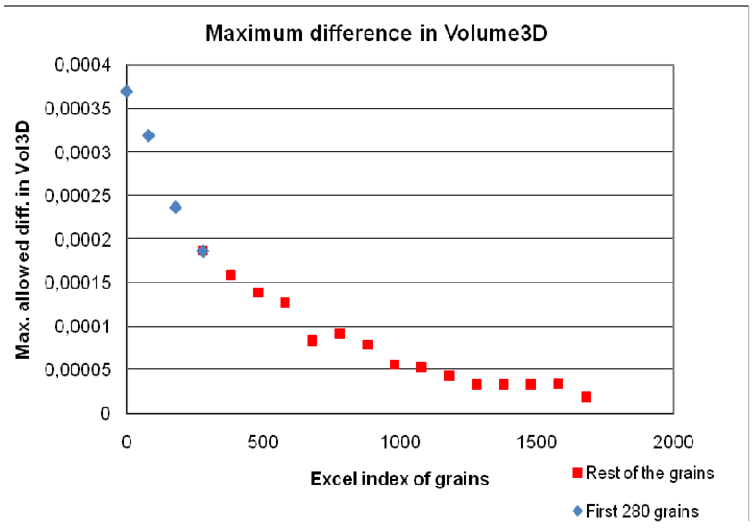


***Figure 20:** This graph shows the decreasing trend of the standard deviation. A trendline through the blue dots represents the linear equation and a trendline through the red dots represents the logarithmic part.*

The filtering based on the difference in volume 3D was also improved. When this selection procedure is not improved, a diminishing amount of results is obtained as the grain size becomes smaller. This is caused by the effect that grains with bigger volumes also tend to have larger differences in volume3D when they are compared to their corresponding alter ego.

Instead of choosing a fixed value, now a variable function was created for the maximum allowed difference in Volume3D. The main requirement was that every bin of 100 consecutive excel indices needed to have the same amount of results. No success was booked in finding a theoretical relation. Instead an attempt was made to find an empirical relation for the decreasing difference in Vol3D for grains with an increasing Excel index.

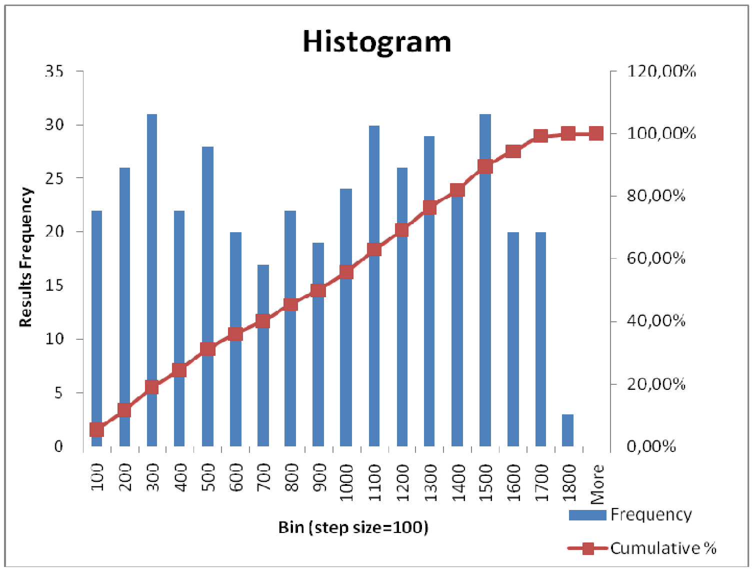


***Figure 21:** This histogram shows the amount of results per 100 grains found with the variable on the difference in Volume3D.*

The formula that was used for the restriction on the difference in volume3D was therefore derived, although not directly, from the standard deviations of every 100 consecutive Excel indices (see figure 20).

With some testing in Matlab, finally two equations were chosen: A linear equation was chosen for the first 300 grains, where the restriction on the difference in Vol3D needs to decrease rapidly in order to find the correct corresponding grains. The grains with an Excel index higher than 300 are filtered on their difference in Vol3D by a logarithmic equation, where the restriction on the difference in Vol3D is decreasing increasingly slower as the Excel indices increase. The results per 100 consecutive excel indices are displayed in figure 21. In total 414 results (1/4 of the total amount of grains observed).

The improved and second data processing method for the rotation test:
The possible reasons for the lack of results on the rotation test in the first attempt will be discussed more thoroughly in chapter 5.2. One of the main reasons why these first results were less than optimal however, can be attributed to the huge increase in the amount of grains that needed to be compared (with regard to the compression test). Even some amplifications were made to the image processing and data processing methods, they were not solid enough to process the data with the precision that is required to apply discrete 3D DIC using particle reconnaissance.

With the initial data processing method, the focus was on the filters that could be applied. Later was thought of a better way of acquiring the correct corresponding grains. This is done by improving the selection procedure of grains in the new Matlab script.

While in the "comparegrains.m" file, corresponding grains were selected if a first result was found, the 'compare_allparameters.m' script first obtains all results for each grain and then selects the best of these results. Let's illustrate this with an example: It can occur that for a single grain 6 matches are found with the implemented filtering criteria. Then all of these results are compared and the best result will be kept. If no results are found for a particular grain, then a row of zeros is entered in the row of the grain in question (meaning no results are found).
This latest m-file is called "compare_allparameters.m" and can be found in Appendix 2.

4.1.3 Rejection of spherical grains:

The measure of the inertia in the angular motion is the moment of inertia. The moment of inertia of a particle or grain is not only related to its mass but also to the distribution of the mass throughout the body. This is how two bodies of the same mass may possess different moments of inertia.

A rigid body like a grain can be considered as a system of particles in which the relative positions of the particles do not change. The moment of inertia of a grain is given by:

$$I = \sum_i m_i r_i^2$$

Now the matrix of moments of inertia or tensor of inertia, J, of a 3D grain with respect to the origin on the reference frame can now be expressed as:

$$J = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{pmatrix}$$

When the particle is aligned with its principal axis, all the off diagonal terms of this matrix become zero.

How this principle was used in for this research may be illustrated with an example. Let's consider a sphere in a 3D space and let's say it is aligned with the local principal axis. When we calculate the moments of inertia of this sphere we will find that all three moments of inertia, which are I_{xx}, I_{yy} and I_{zz}, have equal dimensions. This is, of course, caused by the symmetry of the sphere. As a result we will only find one result when calculating the eigenvalues. With this example in mind we can understand that as a particle (grain) is more spherical, the eigenvalues are closer to one and another.

The eigenvalues of grains tell us how 'streched up' a grain is in a certain direction. Therefore the possibility exists that with spherical grains shifting between eigenvalues occurs.

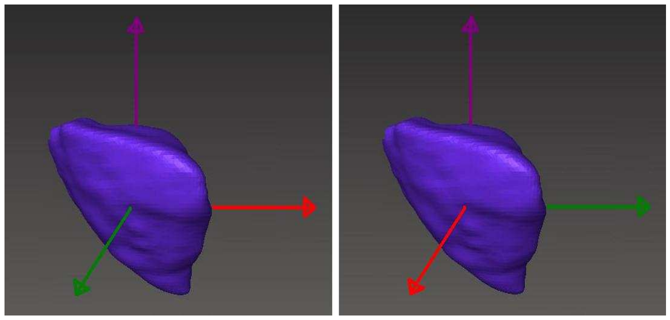


Figure 22: *The grain has not moved but shifting of the principal axis has occurred caused by a small difference in eigenvalues. Corresponding colors are corresponding axis.*

I.e. if the first and second eigenvalue are very close together there is a possibility that these eigenvalues have exchanged positions. (Caused by a small change in volume in the image before and after compression/rotation) If this is the case an erroneous computation of the rotation matrix will be done. For example: if a grain has not rotated at all but the principal eigenvalue and the secondary eigenvalue have shifted, the rotation matrix will rotate the grain 90 degrees in order to align the principal axis of the grain. (see figure 22)

Grains with low differences in eigenvalues should therefore be removed. The question is however, how low this value should be in order to remove the grain based on this given. In this bachelor project no research has been done on this matter. It can be noticed however, that the difference in eigenvalues is (like the difference in Volume3D) decreasing as the Excel indices of the grains increase. During the research the suspicion has arisen that this decrease is non-linear. There is however, no prove to this suspicion yet.
No filtering is done based on the eigenvalues. Instead grains with very high rotation angles have been manually filtered for the compression test. This could however not be done for the rotation test since the rotation angles could be any size.

The application of a filter on grains with low values of Shape_VA3d in Avizo might also be exploited for the cause mentioned above. As mentioned in chapter 3.3 a filter based on this given only selects the well rounded grains. The relation between rounded grains and shifting of eigenvalues should be exploited.

4.2 Threshold difference comparison

Before the data is thoroughly processed the selection of the thresholds needs to be validated. I.e. if the average difference in Volume3D is too big for corresponding grains, fewer grains will be detected. In this case one of the thresholds needs to be adjusted to a level where the volumes of identical grains correspond better.

To test the influence of a difference in grey level threshold on the recognition of grains, an image analysis with 3 different thresholds on a small and controlled volume was done. The first threshold was the supposed optimal threshold; the second was taken a bit larger (55 greylevel steps) and the third a lot smaller (200 greylevel steps).

With the use of Matlab script "comparegrains.m" (Appendix 2) I applied all the thresholds obtained from the manual comparison. In table 3 below it is shown how many grains are found for a certain threshold.

Lower greylevel Threshold	N^o of results	N^o of evidently false results
7745	136	3
7945	340	2
8000	337	1

Table 3: *The transformation required to align the surfaces of the reference grains*

From this table it can be derived that a small difference in thresholding does not by definition lead to bad results. Even though 7945 was assumed to be the optimal lower boundary, a threshold at 8000 leads to almost the same results.
When the threshold is taken much lower however, it does lead to much fewer results. This phenomenon can also be assigned to the increasing 'gluing effect' that increases as the threshold is taken over a larger range. If the lower threshold is taken too low, the contact area between grains will appear bigger than it is. Since grains that have large contact areas are harder to separate, a lot of erroneous separations will be made and grains are harder to detect.

As described in the previous chapter, multiple thresholds were applied on the images of the rotation test in an attempt to bypass the diminishing x-ray effect and create a useful binary image. The cumulative effect of all errors that can be made by applying these different thresholds has not been tested.

4.3 Applications of moments of inertia

The tensor of inertia is a 3x3 matrix that consists of the second order moments of inertia and has different tensors on the diagonal and non-diagonal, respectively, three axial moments of inertia and three products of inertia.
Such a matrix can provide us with complete information about angular acceleration capability for a grain that performs 3D motion. The moments and products of inertia of each grain were obtained from the analysis performed with Avizo. The tensor of inertia J of a grain looks as follows:

$$J = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{pmatrix}$$

With:

$$I_{xx} \equiv \sum_i m_i (y_i^2 + z_i^2) \quad I_{yy} \equiv \sum_i m_i (z_i^2 + x_i^2) \quad I_{zz} \equiv \sum_i m_i (x_i^2 + y_i^2)$$

$$I_{xy} = I_{yx} \equiv -\sum_i m_i x_i y_i$$

$$I_{xz} = I_{zx} \equiv -\sum_i m_i x_i z_i$$

$$I_{yz} = I_{zy} \equiv -\sum_i m_i y_i z_i$$

And for a rigid body, the relative position of particles does not change and these equations can be rewritten to:

$$I_{xx} = \int \left(y^2 + z^2 \right) \cdot dm \quad I_{yy} = \int \left(z^2 + x^2 \right) \cdot dm \quad I_{zz} = \int \left(x^2 + y^2 \right) \cdot dm$$

$$I_{xy} = I_{yx} = -\int xy \cdot dm$$

$$I_{xz} = I_{zx} = -\int xz \cdot dm$$

$$I_{yz} = I_{zy} = -\int yz \cdot dm$$

For the experiments Matlab was used to determine the eigenvalues and eigenvectors from the tensor of inertia. Now follows a brief overview of how these are calculated:

$$Jx = \lambda x$$

In which:
λ = eigenvalues
x = eigenvectors

This can be rewritten to the following formula:

$$\det[J-\lambda \mathrm{I}] = -\lambda^3 + \lambda^2 * tr(J) + \lambda \frac{1}{2}[tr(J^2) - tr^2(J)] + \det(J)$$

det = determinant
λ = eigenvalue
tr = the trace (sum of the elements on the main diagonal) of J
I = identity matrix

Consequently the eigenvectors of the grain can be found by solving the following equation for all three eigenvalues.

$$(J-\lambda \mathrm{I})\mathrm{x}=0$$

The eigenvectors of each corresponding eigenvalue are placed in the same column and the result is the 3x3 vector matrix (P) of the grain. Since we want to describe the way grains have moved, it is necessary to find the 3x3 rotation matrix ($T_{B/A}$) for each grain. With the use of this matrix, a grain can be rotated from its initial state (P_A) to its new state (P_B).

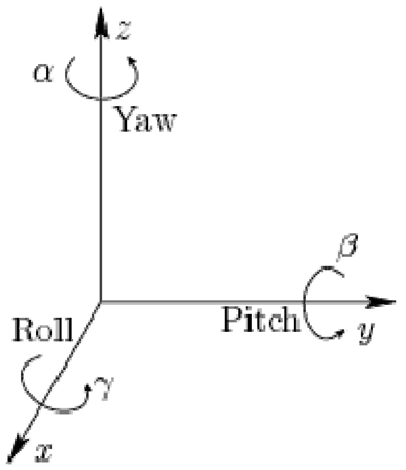


Figure 23: *Any 3D rotation of a grain can be described as a sequence of Yaw, Pitch and Roll rotations*

$$P_B = T_{B/A} * P_A$$

This can be rewritten to:

$$T_{B/A} = P_A^{-1} * P_B$$

Every rotation of a grain, or for that matter any other object in a given volume, can be described as a series of rotations about the 3 global axes. The rotation around the x-, y- and z-axis are called respectively Roll (γ), Pitch (β) and Yaw (α). (see figure 23) So now the transformation can also be written as a function of α, β and γ:

$$T_{B/A}(\alpha,\beta,\gamma) = R_z(\alpha) * R_y(\beta) * R_x(\gamma)$$

$$T_{B/A}(\alpha,\beta,\gamma) = \begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{pmatrix}.$$

From this we can calculate the yaw, pitch and roll.

Let $T_{B/A}(\alpha,\beta,\gamma) = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$, then:

$$\alpha = \tan^{-1}(r_{21}/r_{11})$$
$$\beta = \tan^{-1}(-r_{31}/\sqrt{r_{32}^2 + r_{33}^2})$$
$$\gamma = \tan^{-1}(r_{32}/r_{33})$$

It is of importance that the transformation $T_{B/A}(\alpha,\beta,\gamma)$ is executed correctly. The right order is first roll, thereafter pitch and then yaw as last rotation [Ref 1]. If the order of these operations is changed, a different transformation matrix is obtained.

Rotation is a rigid body movement where, unlike translation, one point is kept fixed. This datum poses us a problem for the rotation of grains in Avizo: the obtained rotation matrices are rotations that occur around the world centre and not around the gravity centre of the grain. Therefore an additional translation of the grain is required in order to undo the movement caused by the rotations around the world axis. (see figure 25 on the next page)

For the compression as well as for the rotation test, the transformation matrix and the yaw, pitch and roll were computed with the Matlab script transform.m (see Appendix 2).

What was the case for the Rubik's Cube in the example of figure 25, also accounts for the grains in Avizo. When a rotation is performed on a grain, additional translation needs to be applied to correct for the movement around the world axis.

This 'additional translation' may be applied together with the normal translation after the rotation sequence is performed. By 'normal translation', it is meant that the translation that takes in account the 3d movement of the grain from its initial position before compression/rotation to its new position (see figure 25).

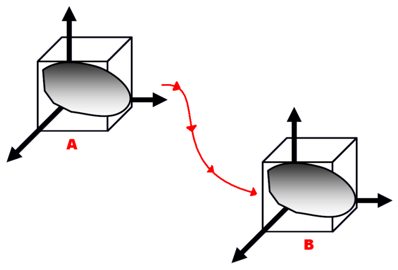
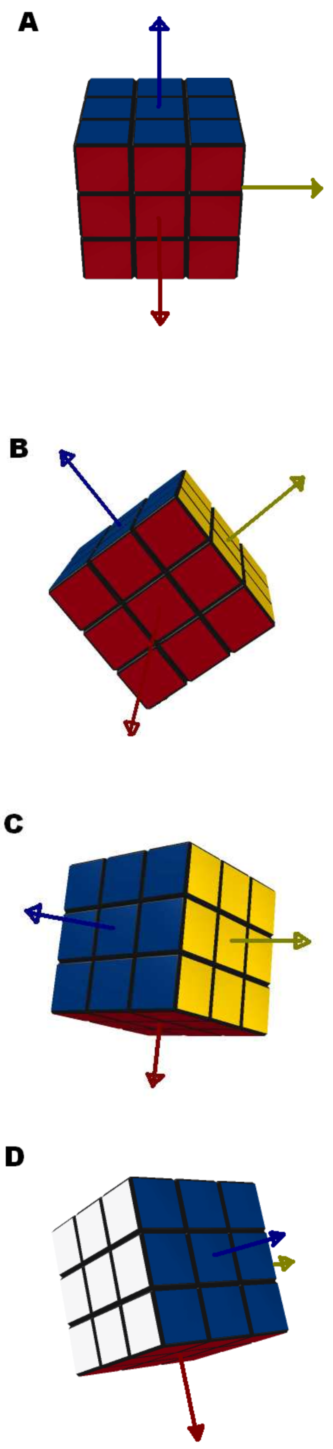


Figure 24:
Translation from a grain in its initial position A, to its new position B.

$$\begin{pmatrix} X_B \\ Y_B \\ Z_B \end{pmatrix} = T_{B/A} * \begin{pmatrix} X_A \\ Y_A \\ Z_A \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

With $T_{B/A} * \begin{pmatrix} X_A \\ Y_A \\ Z_A \end{pmatrix}$ = additional translation and $\begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$ = normal translation.



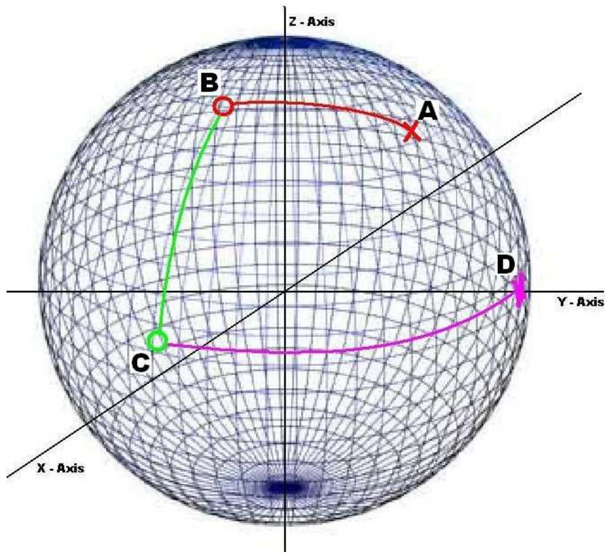


Figure 25:

With the help of these images I want to clarify two things:

1) *The rotation that is being performed on an object is independent of its body frame axis. Instead the object is rotated around the world axis.*
2) *After the object has been rotated around the world axis, it is on a different place in the coordinate system than it initially was.*

On the left four orientations of a Rubik's Cube, R_c, in a volume $\mathbb{R}^3$ are considered. From position A, its original position, consecutively roll, pitch and yaw are performed on the cube. The position the cube is in after each rotation is respectivily B, C and D.

A rotation simply rotates every point in $\mathbb{R}^3$ in terms of the world frame. The blue grid sphere in the image above therefore visualizes the plane the cube can move in when it is being rotated. On the grid, the trajectory of each successive rotation step is displayed:

A-B= Roll
B-C= Pitch
C-D= Yaw

A translation from position D to position A needs to be applied if we want to align the centre of gravity of R_c before and the centre of gravity of R_c after rotation.

4.4 Obtaining the rotation angle for Avizo

In Avizo the rotations are applied with the use of the axis angle and corresponding increments for the rotation around each axis.
A 3×3 rotation matrix has 9 components and it contains replicated information ($r_{12} = r_{21}$ etc.). There are a lot of ways to derive the rotation from the numbers; this is the way that was used to convert the rotation matrix to axis angle with corresponding increments for the rotation around X- Y- and Z-axis. [ref. 2]

$$\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \text{Rotation matrix}$$

$\text{angle} = \text{acos}((r_{11} + r_{22} + r_{33} - 1)/2)$
$x = (r_{32} - r_{23}) / \sqrt{((r_{32} - r_{23})^2 + (r_{13} - r_{31})^2 + (r_{21} - r_{12})^2)}$
$y = (r_{13} - r_{31}) / \sqrt{((r_{32} - r_{23})^2 + (r_{13} - r_{31})^2 + (r_{21} - r_{12})^2)}$
$z = (r_{21} - r_{12}) / \sqrt{((r_{32} - r_{23})^2 + (r_{13} - r_{31})^2 + (r_{21} - r_{12})^2)}$

There are two singularities at angle = 0° and angle = 180°, in these cases the above formula may not work. A solution for this problem has not been implemented. From manual inspection of the results that have been obtained however, not once an occurrence of the angle being exactly 0 or 180 degrees has been noticed.

The corresponding axis angle has successfully been found when manual rotation was applied on a grain (consequently around the x-axis, y-axis and z-axis). So in this case the actual roll, pitch and yaw performed on the grain were known and used to compute the axis angle. This did however not work when the axis angle was derived in the way described above. I.e. when the moments of inertia are used to compute the rotation matrix and the axis angle is obtained from this matrix.

This mistake in the computation remains unsolved but it is suspected that the core of the problem lies either in the way the BinMom's (=moments of inertia) are defined in Avizo or in the way the transformation matrix is computed in Matlab.
In consultation with Dr. Ir. D. Ngan-Tillard it was assumed that although no success was booked in rotating a grain in Avizo, the order of magnitude of the yaw, pitch and roll was still correct. Therefore the yaw, pitch and roll obtained from the rotation matrix was used to visualize the rotation of grains in Avizo.

4.5 Filter creation

If a range of grains needs to be selected in Avizo, a filter of some kind needs to be created to filter the required grains from the debris. In order to create a filter, first a range of values needed to be selected in Excel. To decide what range to choose, an online tool [ref.3] was used that determines the optimal bin size. With the compression test, this is also done for the displacement in the z-direction (Figure 26). According to this histogram the best distribution exists of four different ranges in the z-direction.

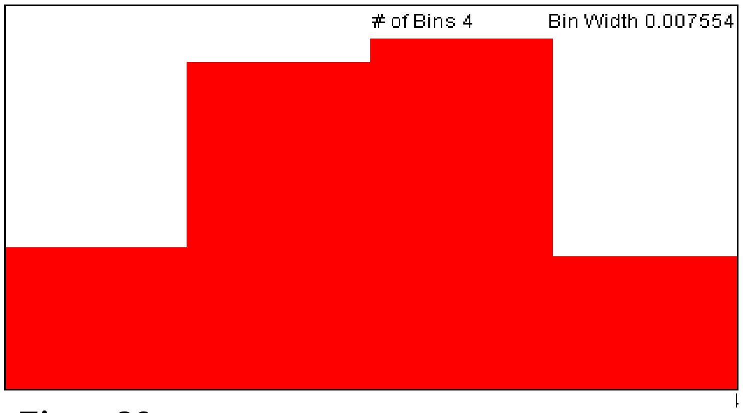


Figure 26:
This figure shows the suggested bin size for the displacements of grains in the z-direction.
Image obtained from [ref.5]

To prevent drudgery, a script was written with Visual Basic (appendix 3) that can select multiple rows based on a value in a single cell. This program was used to select and sort the shape parameters of the grains based on the displacement in the z-direction according to the suggested bins.

At the time this research was performed, it was not possible to add values from Excel to the result viewer in Avizo. In order to be able to create a 3D image in Avizo, a filter was created in Excel based on the Area3D and the Volume3D such that this filter was unique for the referred grain.

```
OR ((( Area3d >     X - 0.0000001    ) AND ( Area3d <     X + 0.0000001    )) AND ((Volume3d >
  Y - 0,0000001    ) AND ( Volume3d <       Y + 0,0000001    )))
```

With the variable parameters:

X = The amount of difference in Area3d
Y= The amount of difference in Volume3d

Following the same procedure, filters for the rotation test were created. Filters for all bin's were created manually in Avizo. It is however also possible to write a script that will do just that.

5 Results

5.1 Compression test

For the compression test one would expect to see the largest vertical displacements on the upper side of the measured volume and the lowest amount of vertical displacement on the bottom.

This is also what the images in Avizo depict as illustrated in figures 27 and 28. The x- and y-axis are respectively red and green. The vertical z-axis is colored blue. The corresponding grain colors are:

Red:		displacement	< 0.12 mm
Orange:	0.12mm <	displacement	< 0.13mm
Yellow:	0.13mm <	displacement	< 0.14mm
White:		displacement	>0.14 mm

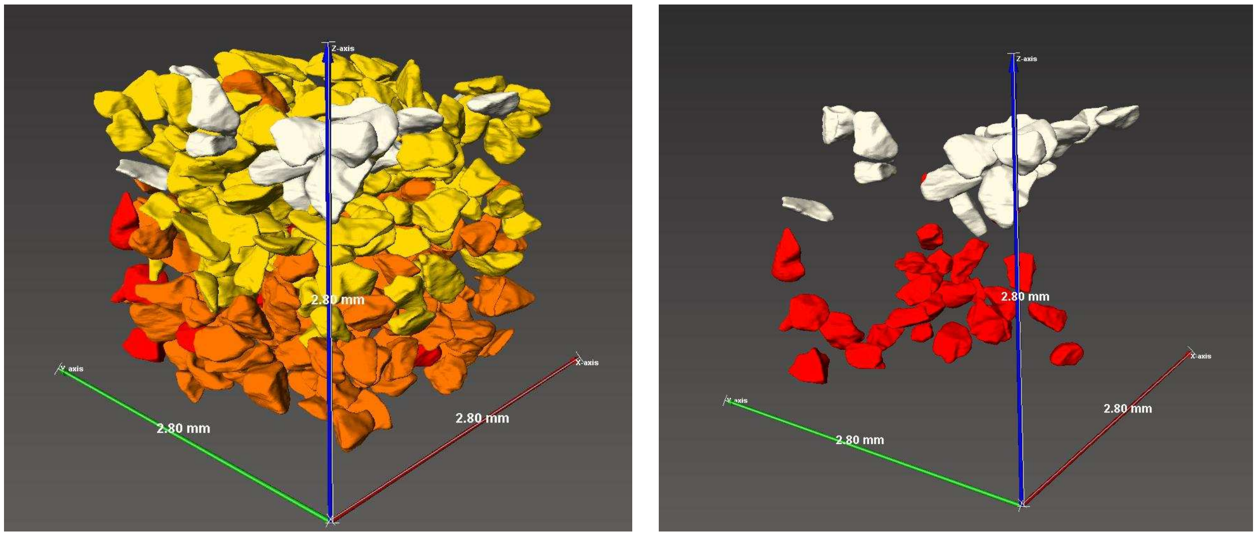


Figure 27: *Left, 3D image of all grains. Right, 3D image where only grain with a relatively high (white) and low (red) displacements are shown. Max. displacement = 0.155 and min. displacement = 0.114*

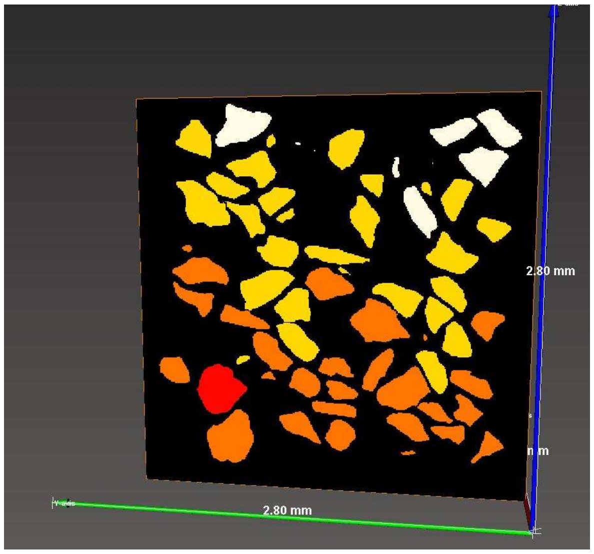
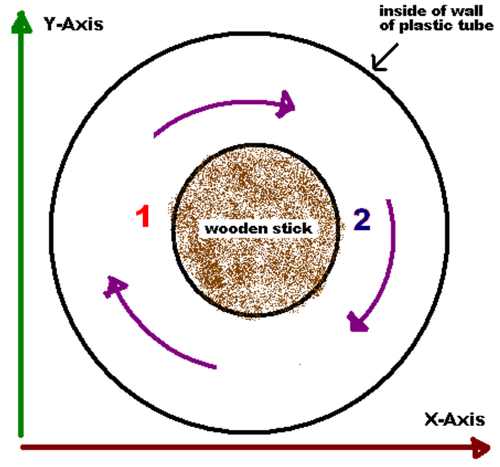


Figure 28: *2D OrthoSlice in the YZ-plane*

5.2 Rotation test

With the rotation test, a clockwise rotation was performed on the wooden stick in the research mounting. The idea of this is that irregularities on the wooden stick cause the grains to move. One would expect to see that the grains that are closest to the wooden cylinder, before it is rotated, perform the highest amount of translation and rotation. The grains that are further away from the centre of the mounting should then appear have remained more stationary.

As was mentioned in Chapter 4.1.2, the data of the rotation test has been analyzed twice because of unexpected results after the initial analysis. This chapter will first discuss the first (erroneous) results, next the errors in the analyzing process will be described and finally the second, improved results on the rotation test will be discussed.



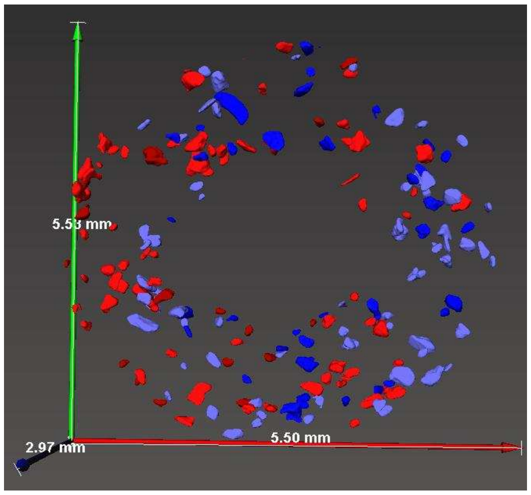
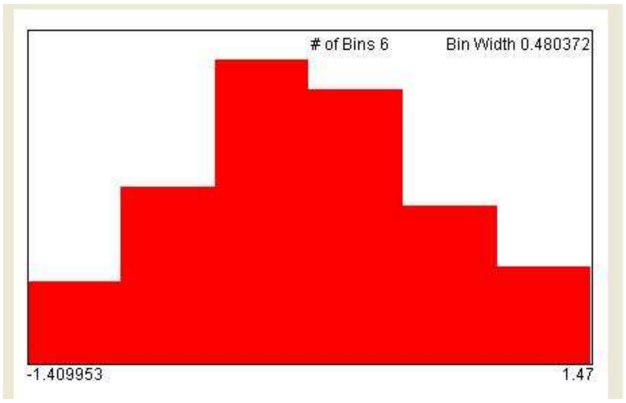


Figure 29A and 29B:
Left: schematic overview of the research mounting and the rotation performed by the wooden tooth stick. Locations 1 and 2 point out where I would have expected to find respectively the grains with largest negative and the grains with the largest positive displacement in the x-direction.
Right: a 3D visualization of the grains that have performed the highest amount of displacement in the x-direction according to the results obtained from the data processing. (red: disp > 0.5 mm and blue: disp < -0.5 mm)



The distribution of the displacement in the x-direction is displayed in the histogram below (figure 30). The places where one would expect to find grains that have performed the highest amount of displacement in the x-direction are shown in the schematic overview of figure 29A.

Figure 30: *histogram of the displacement in x-direction.* Image obtained from [ref.5]

With the use of the filters obtained from the data processing (which has been explained in chapter 4), a 3D image in Avizo was obtained. In the resulting image (figure 29B) , only the grains with the largest displacements in the x-direction are shown.

A conclusion not hard to draw from figure 29B is that the result is not what one would expect from theoretical reasoning. It is believed that the following five causes might have contributed to a poor result:

1. The step size of the multiple thresholding, which was applied to correct for the decrease in greylevel intensity towards the center. Four cylinders with a consecutive increasing threshold-span (as the diameter of the cylinder decreases), were applied. These increases in threshold-span were quite abrupt. The span was increased by 50 greylevel points with each step. This might have caused erroneous grain segmentation, as grains on the border of two threshold-spans appear to 'thin out' towards the smallest threshold span.

2. Although the effect of small differences in threshold-span on the amount of corresponding grains found was tested (chapter 4.2), this effect might be different in the case that multiple thresholds are applied on one image.

3. For the rotation test smaller grains were used than for the compression test. No extensive research has been done to the best segmentation procedure for these smaller grains. Basically the segmentation process from the compression test has been adopted for the rotation test as it had also worked quite well for the somewhat bigger grains of the first test.

4. Even though a smaller volume is considered with the rotation test than was the case for the compression test, a lot more grains are observed because of the smaller diameter of grains. As discussed in chapter 4, the shape parameters are therefore closer together which makes the data processing more difficult.

5. The way particle reconnaissance is performed on the data of the rotation test is therefore entirely different from and more complex than the particle reconnaissance done for the compression test.

The results of the second attempt to analyze the data from the rotation test were somewhat better. This time, no multiple thresholding was applied. To make the segmentation process easier, only a small volume of grains was considered very close to the wooden stick. In figure 31 below, the X- and Y- displacement of these grains and the corresponding histograms are shown. The x- and y-axis are respectively red and green. The vertical z-axis is colored blue. The corresponding grain colors for the X-displacement:

Red:		displacement	> 2.00 mm
Lightred:	1.00mm <	displacement	< 2.00 mm
Lightred-white:	0.00mm <	displacement	< 1.00 mm
White-Lightblue:	-1.00mm <	displacement	< 0.00 mm
Lightblue:	-2.00mm <	displacement	< -1.00 mm
Blue		displacement	< -2.00 mm

The corresponding grain colors for the Y-displacement:

Red:		displacement	> 1.30 mm
Lightred:	0.40mm <	displacement	< 1.30 mm
White:	-0.40mm <	displacement	< 0.40 mm
Lightblue:	-1.30mm <	displacement	< -0.40 mm
Blue		displacement	< -1.30 mm

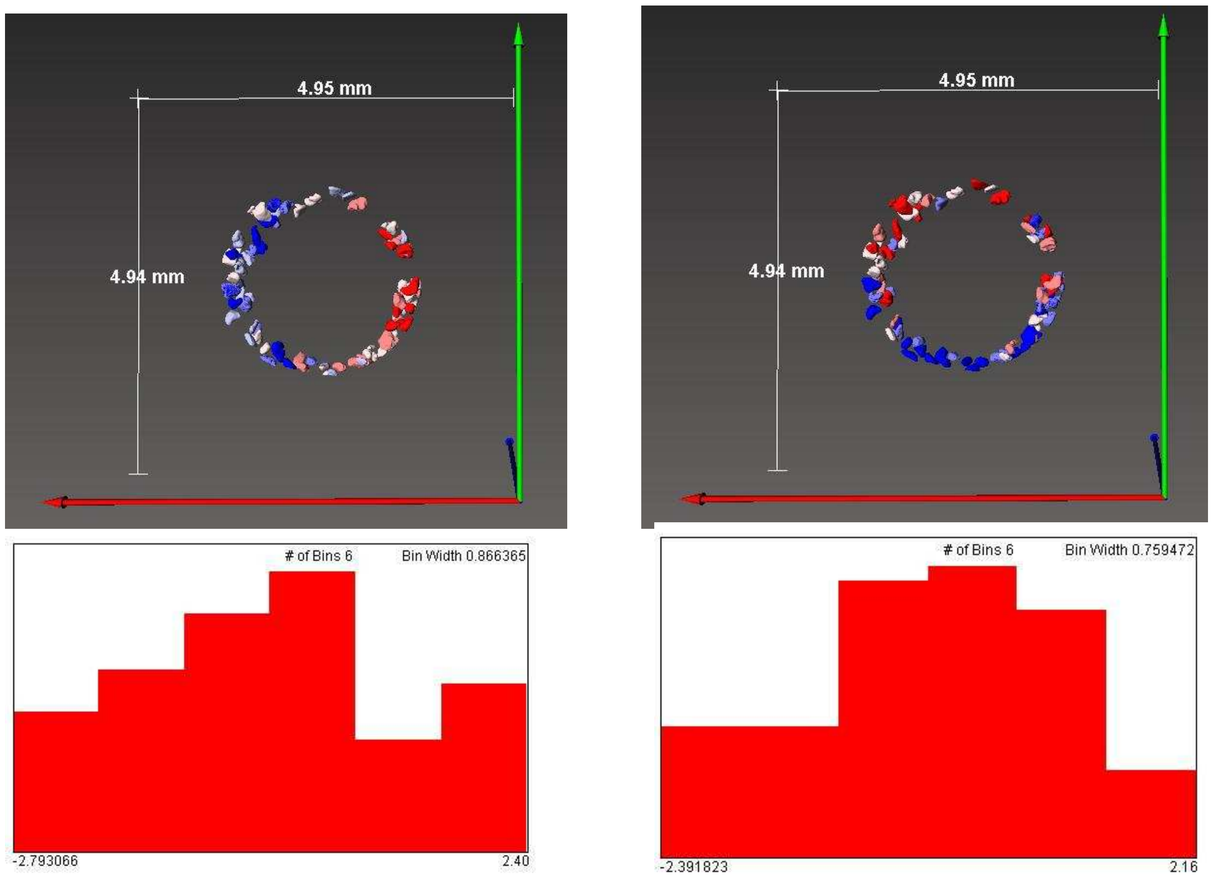


Figure 31: *X- and Y- displacement of grains by rotation. 83 grains are shown.*

Because we are only looking at the innermost circle of grains, the rotation is harder to analyze. As we expect that the amount of rotation for grains diminishes as the grains are further from the centre of the mounting. In figure 32 the distribution of the roll, pitch and yaw for the grains is shown. Figure 33 shows the assumed yaw of grains. When more 'threshold layers' (i.e. wider circles around the centre) are also analyzed, perhaps a diminishing yaw can be observed as the grains are situated more outwards. Slightly more negative rotation (blue grains) is observed as can be seen in figure 33, which does correspond with a clockwise rotation. There is however to little valid information at hand to verify whether this 3D image really shows us the proper rotation of grains around the world-z-axis.

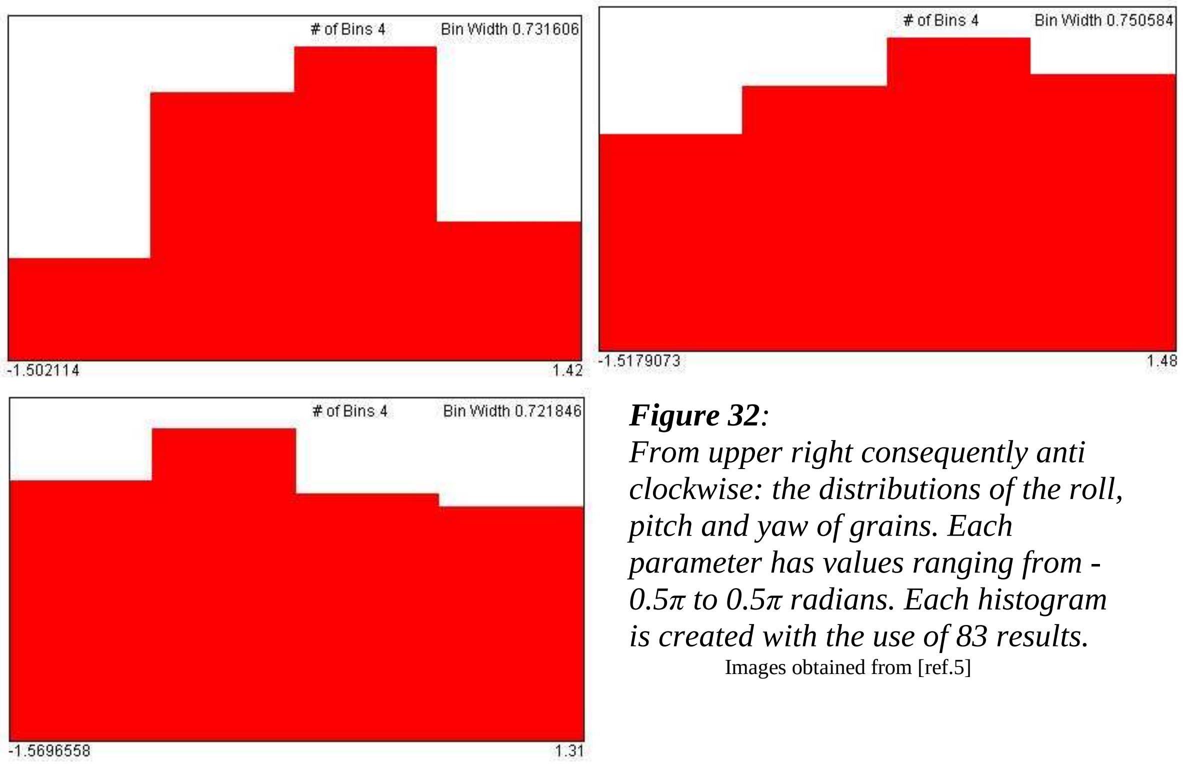
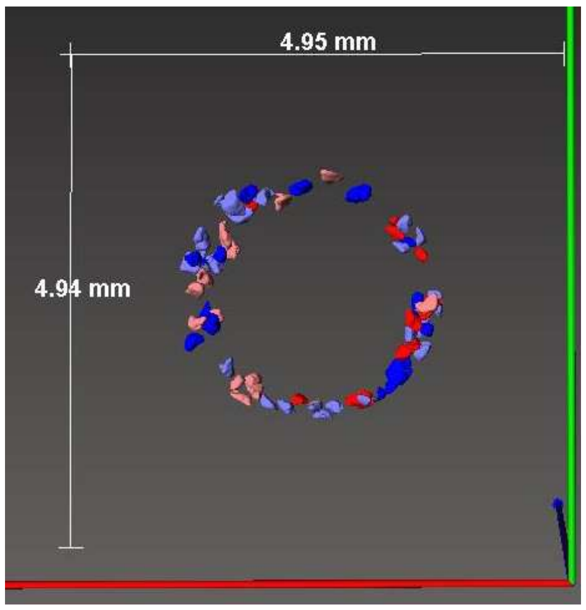


***Figure 32**:*
From upper right consequently anti clockwise: the distributions of the roll, pitch and yaw of grains. Each parameter has values ranging from -0.5π to 0.5π radians. Each histogram is created with the use of 83 results.
Images obtained from [ref.5]



Firgure 33:
Yaw of grains. The z-axis is pointing into the paper. Blue is -1 to- 0.33 radians; light blue to light red is -0.33 to 0.33 radians and red is 0.33 to 1 radians.

6 Conclusions and recommendations

Up to 90 % of the particles in a research mounting can be refound using particle reconnaissance. With a retrieval percentage this high, good images can be made of discrete media. Therefore we can conclude that particle reconnaissance has the potential to be a solid method to apply for creating future discrete 3D DIC images. When this method is further improved it should be possible to localize strain quite accurately in discrete media using 3D DIC.
One major drawback with particle reconnaissance is the time elapse that is required to process the image, find corresponding grains and create filters. If it would be possible to automate these processes, this processing method could prove to be useful for future 3D DIC. In the rest of this chapter the bottlenecks that were encountered during this research will be discussed.

Improvements on the image processing of the grains:
Even though the use of Maas sand worked quite well, it is recommended to use pure quartz grains for future development of discrete 3D DIC methods. The use of 100 % quartz would probably simplify the segmentation process as the differences between grains and pores can be made more evident when the contrast among grains decreases.

It will prove to be useful if the two images that need to be compared, have the same grey scale. While it was not sure (yet) how the CT-scanner addresses the greylevels to the grains and why the greylevels differ with each consecutive test, it might be an option to calibrate the gray scales with the use of a material in the research mounting as reference grey scale. This calibration material could for example be the casing of the tube.

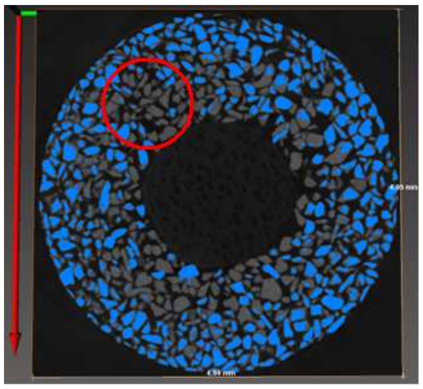


Figure 32: *Unequal beam hardening*

Smaller grains should be attached to the tube as a reference to reduce the unequal ‘beam hardening’ effect: If reference grains are too large, they will ‘radiate’ and cause certain grains to have lower greylevels than their neighbors. In Figure 32 a section with locally lower greylevels has been circled.

The thresholding process is very important and the key factor to a good result. Although it has been proved in chapter 4.1 that a small fluctuation in threshold does not have to affect the amount of grains that is detected, it has not been checked if this also true for the case where multiple thresholds are applied (this was done for the rotation test to diminish the beam hardening effect). The risk of this method is that grains can be digitally deformed if they lie on the border of two thresholds. In that case the grain will be thinner on the side where a smaller threshold range is applied.
The effect of this should be evaluated. One way of doing this is reducing the step size by which the threshold is reduced with each cylinder (for example 5 greylevels per cylinder instead of 50). This would then require more separate threshold cylinders, which would take

a long time to perform manually. Perhaps automation by writing a script for this would reduce the time elapse required.
Still, it has been proved that it is possible to perform discrete 3D DIC using particle reconnaissance on the innermost threshold circle of the rotation test with the latest scripts. When consequent analysis on each threshold ring is done it should therefore also be possible to create a better image of the Yaw of grains. When this is done, it can be expected that the yaw of grains diminishes when the grains are located further away form the centre of the research mounting.

The way transformations on grains can be applied in Avizo needs to be figured out. At this point I have not figured out a way to properly apply the required transformations on grains in Avizo. Gwenole Tallec, R&D senior software engineer of VSG pointed out in a mail that 'a lot of information from the TCL console (see SpatialData documentation) can be obtained (getTranslation, getTransform,…)'.

Improvements on the selection and filtering process:
The grain selection procedure (data processing) that was used to analyze the grains from the compression test and initially also the rotation test, was not optimal. But the method worked for the processing of the data of the compression test as was discussed in chapter 4.
The latest selection procedure that was developed to recognize and select grains of the rotation test is more refined and precise. Based on the results and the images that were obtained for the innermost threshold circle of the rotation test, it can be concluded that by first selecting all suspected pairs of grains and then choosing the best result, gives good results.

With regard to the eigenvalue shifting, discussed in chapter 4.1.3, filtering based on differences in eigenvalues should be exploited. The relation of small values of Shape_VA3d to small difference in eigenvalues should also be considered. Perhaps it is possible to filter the grains in question early in the process by applying a filter on low Shape_VA3d values in Avizo.

A drawback on the way this method of 3D DIC is being performed is that a lot of different programs are required to perform a good selection of grains in Avizo. Smarter programming and automating even more processes would speed up the process a lot.

References

Ref 1: Planning Algorithms, Steven M. LaValle, available online at:
http://planning.cs.uiuc.edu/node91.html
Ref 2:
http://www.euclideanspace.com/maths/geometry/rotations/conversions/matrixToAngle/
Ref 3: http://www.ton.scphys.kyoto-u.ac.jp/~hideaki/res/histogram.html
Ref 4: Linear Algabra and its applications (third edition), David C. Lay
Ref 5: http://www.kwon3d.com/theory/transform/transform.html
Ref 6: http://www.randelshofer.ch/rubik/virtualcubes/virtual_rubik_en.html

Appendix 1: Avizo Scrips

Index:

These scripts were used for the compression test. Here the threshold is a variable. After these scripts were applied, the filters on Shape_VA3d and the volume borders were implemented manually. As a starting point for these scripts, the script from the 'mousse'-example in Avizo 6 was used.
The same procedure was executed for the rotation test but no script was used. For this test multiple threshold ranges were applied manually in an attempt to diminish the 'beam hardening-effect'.

Script first part

This script was used to perform the first few steps of the image processing.

```
###############################

# constructor is called when scro is instantiated
$this proc constructor {} {

    remove Visilog:algo

    $this script hide
    $this newPortDoIt DoIt
    $this DoIt setLabel 0 "DoIt"
    $this DoIt setLabel ""
    $this DoIt setCmd 0 {$this computeDoIt}
}


# constructor is called when scro is instantiated
$this proc destructor {} {

}

$this proc computeDoIt {} {
```

```
# redraw for the user
viewer redraw

set data [$this data source]

if {$data == ""} {return}

  create HxVisilog {Visilog:algo}
  # Visilog:algo hideIcon
  Visilog:algo Execution setValue 0

  $this select

  # enable stop button
  workArea startWorking Cafe-Demo

  # put no_calib as current calib
  Visilog:algo exeCommand {cmd=currentcalibration param=no_calib param=no}

  echo "filter"
  set firstCmd "cmd=medianfilter3d input=$data param=3 output=medianf"
  Visilog:algo exeCommand $firstCmd

  echo "threshold"
  Visilog:algo exeCommand {cmd=threshold input=medianf param={9350,20000} output=th }
  #The threshold is a variable

  echo "hole_fill"
  Visilog:algo exeCommand {cmd=hole_fill input=th param=3D_6 output=thhf }

  if {[workArea wasInterrupted]} {
        workArea stopWorking
        return
  }

  #Separation By Maxima distance
  ###############################

  # prendre les maxima dilatés comme marqueurs
  echo "maxima dilatés"
  Visilog:algo exeCommand {cmd=distance input=thhf output=ima_dist}
```

```
#       Visilog:algo exeCommand {cmd=distxxx input=thhf param=1 param=1.41 param=1.73 output=ima_dist}
        Visilog:algo exeCommand {cmd=merge_maxima input=ima_dist param=1 output=marqueurs}
        # marqueurs hideIcon
        Visilog:algo exeCommand {cmd=label input=marqueurs output=marqueurs2}

        if {[workArea wasInterrupted]} {
                workArea stopWorking
                return
        }

        #inversion de la distance
        echo "logical_not"
        Visilog:algo exeCommand {cmd=logical_not input=ima_dist output=ima_dist2}

        # bassins versants et separation
        echo "bassins versants"
        Visilog:algo exeCommand {cmd=fastwatershed input=ima_dist2 input=marqueurs2 output=wsl}
        Visilog:algo exeCommand {cmd=logical_sub input=thhf input=wsl output=sepgrains}

        remove ima_dist
        remove ima_dist2
        remove marqueurs
        remove marqueurs2

        #############################
        echo "show contacts"
        Visilog:algo exeCommand {cmd=logical_sub input=thhf input=sepgrains output=contacts}
        Visilog:algo exeCommand {cmd=label input=contacts output=contactslb}
        Visilog:algo exeCommand {cmd=I_analyze input=contactslb input=contactslb param=adjust param=Merge analysis
outanl=contactanalysis}
# The “merge analysis” feature has to be created manually and needs to contain the parameters required for manual
separation filtering

        workArea stopWorking
}
```

Script second part

This script was used to perform the steps following the manual removal of erroneous separations.

```
###############################

# constructor is called when scro is instantiated
$this proc constructor {} {

    remove Visilog:algo

    $this script hide
    $this newPortDoIt DoIt
    $this DoIt setLabel 0 "DoIt"
    $this DoIt setLabel ""
    $this DoIt setCmd 0 {$this computeDoIt}
}


# constructor is called when scro is instantiated
$this proc destructor {} {

}

$this proc computeDoIt {} {

    # redraw for the user
    viewer redraw

    set data [$this data source]

    if {$data == ""} {return}

       create HxVisilog {Visilog:algo}
       # Visilog:algo hideIcon
       Visilog:algo Execution setValue 0

       $this select

       # enable stop button
       workArea startWorking Cafe-Demo
```

```
	# put no_calib as current calib
	Visilog:algo exeCommand {cmd=currentcalibration param=no_calib param=no}

	echo "improved wsl"
	set firstCmd "cmd=logical_sub input=wsl input=$data output=wsl_filtered"
	Visilog:algo exeCommand $firstCmd
	# th hideIcon

	Visilog:algo exeCommand {cmd=logical_sub input=thhf input=wsl_filtered output=sepgrains_filtered}

	# labelisation
	################
	echo "labelisation"
	Visilog:algo exeCommand {cmd=label input=sepgrains_filtered output=sepgrainslb}

	if {[workArea wasInterrupted]} {
		workArea stopWorking
		return
	}

	  # creation de l'analyse
	########################
	echo "creation de l'analyse"
	Visilog:algo exeCommand {cmd=I_analyze input=sepgrainslb input=sepgrainslb param=adjust param=BSc project
outanl=grainanalysis}

	workArea stopWorking

}
```

Appendix 2: Matlab Scripts

Index:

Comparegrains.m

This script compares grains from before and after the compression and also calculates the difference in volume and movement in all directions if two grains correspond. Finally it also removes all zeros from the matrix to obtain only the grain numbers for which a result is found.

```
function C = comparegrains(A,B)
%%To compare and discard grains and calculate movement
% A and B need to have rows of the following composition:
% [ (Volume3D) (BaryCenterX) (BaryCenterY) (BaryCenterZ) (Excel index) ]
[m] = size(A);
[n] = size(B);
C = zeros(m(1,1),6);
for i = 1:m(1,1)
    for j = 1:21
           k=i+j-12;
        if and(k <= n(1,1), k > 0) %Determination of matrix borders
            if B(k,1) > 0 %reject rows that have been used before
                if and(abs(A(i,1) - B(k,1)) <= 0.0005, sqrt((abs(A(i,2)-B(k,2)))^2+(abs(A(i,3)-B(k,3)))^2) < 0.051)
%Condition that the value for the difference in Vol3D needs to be smaller than a value manually obtained from
comparison of grains in excel. The second condition is a maximum on the amount of movement in the XY-plane.
                    C(i,1) = A(i,5);
                    C(i,2) = B(k,5);
                    C(i,3) = A(i,1)-B(k,1);
                    C(i,4) = A(i,2)-B(k,2);
                    C(i,5) = A(i,3)-B(k,3);
                    C(i,6) = A(i,4)-B(k,4);
                    B(k,:) = 0; %If i want to compare the matrix with more variables, this row must not be available
since it already has a match.
                    %disp([i j k]);
                    break;
                else
```

```
                    if j == 21
                        C(i,1) = A(i,5);
                        C(i,2)=0;C(i,3)=0;C(i,4)=0;C(i,5)=0;C(i,6)=0;%If there is no row in matrix B which can be
compared to a row from matrix A according to the conditions, then the row in the outputmatrix becomes C=0.
                        %disp([i j k]);
                    else
                    end
                end
            else
                if j == 21
                    C(i,1) = A(i,5);
                    C(i,2)=0;C(i,3)=0;C(i,4)=0;C(i,5)=0;C(i,6)=0;
                    %disp([i j k]);
                else
                end
            end
        else
            if j == 21
                C(i,1) = A(i,5);
                C(i,2)=0;C(i,3)=0;C(i,4)=0;C(i,5)=0;C(i,6)=0;
                %disp([i j k]);
            else
            end
        end
    end
end
disp(C(:,1));
disp(C(:,2));
disp(C(:,3));
disp(C(:,4));
disp(C(:,5));
disp(C(:,6));

%now starts the script to remove the zeros from the matrix (C becomes S)
Q = C(:,3);
[j] = find(Q);
[p,q] = size(j);
S = ones(j,5);
for k = 1:p
    S(k,:) = C(j(k,1),:);
end
```

Transform.m

With the use of this script the rotation matrix of a grain from its initial state to its new state is calculated. Before this can be done, first a list of the numbers of corresponding grains needs to be available.

```
function VenE = transform(J,K,L)
%J: The columns Exel indices of the grains before and after movement
%K: in K(:,1) the excel index and in the remaining 6 columns the BinMom
%parameters from the grains before movement
%L: in L(:,1) the excel index and in the remaining 6 columns the BinMom
%parameters from the grains After movement

%-----------------------------------------------------------------
%Here two matrices (k and l) are created with the values for the BinMom's of the grains resp. before and after movement
%-------------------------------------------------------------------------
p = size(J);
y = size(L);
x = size(K);
l = zeros(p(1,1),7);
k = zeros(p(1,1),7);
for r = 1:p(1,1)
    for s = 1:y(1,1)
        if J(r,2) == L(s,1) % With J the matrix with the right grain numbers and L, the After-matrix containing the
BinMom properties of grains.
            l(r,7) = J(r,2); % This is the Excel index for the after - grain.
            l(r,1) = L(s,2); % BinMom2X
            l(r,2) = L(s,3); % BinMom2Y
            l(r,3) = L(s,4); % BinMom2Z
            l(r,4) = L(s,5); % BinMomXY
            l(r,5) = L(s,6); % BinMomXZ
            l(r,6) = L(s,7); % BinMomYZ
            break;
        else
        end
    end
end
for r = 1:p(1,1)
    for s = 1:x(1,1)
        if J(r,1) == K(s,1) %K: the Before-matrix.
            k(r,7) = J(r,1);
            k(r,1) = K(s,2);
            k(r,2) = K(s,3);
            k(r,3) = K(s,4);
            k(r,4) = K(s,5);
            k(r,5) = K(s,6);
            k(r,6) = K(s,7);
            break;
        else
        end
    end
end
%-----------------------------------------------------------
%Now the eigenvalues and eigenvectors are being calculated
%-----------------------------------------------------------
mat=zeros(3,3);
P0=zeros(p(1,1),10);
P0i=zeros(p(1,1),10);
D0=zeros(p(1,1),4);
P1=zeros(p(1,1),10);
P1i=zeros(p(1,1),10);
D1=zeros(p(1,1),4);
for b = 1:p(1,1) % With this for-loop the eigenvectors/values for the before-grains are calculated.
    mat(1,1) = k(b,1); mat(1,2) = k(b,4); mat(1,3) = k(b,5);
    mat(2,1) = k(b,4); mat(2,2) = k(b,2); mat(2,3) = k(b,6);
    mat(3,1) = k(b,5); mat(3,2) = k(b,6); mat(3,3) = k(b,3);
    [Vuncomp,D] = eig(mat); % The columns of V are the eigenvectors of this grain and D is the diagonal matrix with the
eigenvalues.
    Vuncompi = inv(Vuncomp);
    P0(b,1) = k(b,7);           P0i(b,1) = k(b,7);              %Excel index before.
    P0(b,2) = Vuncomp(1,1);     P0i(b,2) = Vuncompi(1,1);       %P0 is the vectormatrix and P0i the inverse
vectormatrix
    P0(b,3) = Vuncomp(2,1);     P0i(b,3) = Vuncompi(2,1);
    P0(b,4) = Vuncomp(3,1);     P0i(b,4) = Vuncompi(3,1);
    P0(b,5) = Vuncomp(1,2);     P0i(b,5) = Vuncompi(1,2);
    P0(b,6) = Vuncomp(2,2);     P0i(b,6) = Vuncompi(2,2);
    P0(b,7) = Vuncomp(3,2);     P0i(b,7) = Vuncompi(3,2);
    P0(b,8) = Vuncomp(1,3);     P0i(b,8) = Vuncompi(1,3);
    P0(b,9) = Vuncomp(2,3);     P0i(b,9) = Vuncompi(2,3);
    P0(b,10) = Vuncomp(3,3);    P0i(b,10) = Vuncompi(3,3);
    D0(b,1) = k(b,7);                                           %D0 is the matrix with the eigenvalues
    D0(b,2) = D(1,1);
    D0(b,3) = D(2,2);
    D0(b,4) = D(3,3);
```

```
end
for b = 1:p(1,1) %With this for-loop the eigenvectors/values for the after-grains are calculated.
    mat(1,1) = l(b,1); mat(1,2) = l(b,4); mat(1,3) = l(b,5);
    mat(2,1) = l(b,4); mat(2,2) = l(b,2); mat(2,3) = l(b,6);
    mat(3,1) = l(b,5); mat(3,2) = l(b,6); mat(3,3) = l(b,3);
    [V7945,D] = eig(mat);
    V7945i = inv(V7945);
    P1(b,1) = l(b,7);           P1i(b,1) = l(b,7);
    P1(b,2) = V7945(1,1);       P1i(b,2) = V7945i(1,1);
    P1(b,3) = V7945(2,1);       P1i(b,3) = V7945i(2,1);
    P1(b,4) = V7945(3,1);       P1i(b,4) = V7945i(3,1);
    P1(b,5) = V7945(1,2);       P1i(b,5) = V7945i(1,2);
    P1(b,6) = V7945(2,2);       P1i(b,6) = V7945i(2,2);
    P1(b,7) = V7945(3,2);       P1i(b,7) = V7945i(3,2);
    P1(b,8) = V7945(1,3);       P1i(b,8) = V7945i(1,3);
    P1(b,9) = V7945(2,3);       P1i(b,9) = V7945i(2,3);
    P1(b,10) = V7945(3,3);      P1i(b,10) = V7945i(3,3);
    D1(b,1) = l(b,7);
    D1(b,2) = D(1,1);
    D1(b,3) = D(2,2);
    D1(b,4) = D(3,3);
end
p0i = zeros(3,3); p1 = zeros(3,3); rot = zeros(3,3); R7945 = zeros(p(1,1),11);
yaw7945 = zeros(p(1,1),1); pitch7945 = zeros(p(1,1),1); roll7945 = zeros(p(1,1),1);
for b = 1:p(1,1)       % With this for-loop the rotationmatrix and eventually the yaw, pitch en roll are calculated.
    p0i(1,1) = P0i(b,2); p0i(1,2) = P0i(b,5); p0i(1,3) = P0i(b,8);
    p0i(2,1) = P0i(b,3); p0i(2,2) = P0i(b,6); p0i(2,3) = P0i(b,9);
    p0i(3,1) = P0i(b,4); p0i(3,2) = P0i(b,7); p0i(3,3) = P0i(b,10);
    p1(1,1) = P1(b,2); p1(1,2) = P1(b,5); p1(1,3) = P1(b,8);
    p1(2,1) = P1(b,3); p1(2,2) = P1(b,6); p1(2,3) = P1(b,9);
    p1(3,1) = P1(b,4); p1(3,2) = P1(b,7); p1(3,3) = P1(b,10);
    rot=p0i*p1; %Here the rotationmatrix is being computed.
    R7945(b,1)=P0(b,1); %excel index before
    R7945(b,2)=P1(b,1); %excel index after
    R7945(b,3)=rot(1,1);
    R7945(b,4)=rot(2,1);
    R7945(b,5)=rot(3,1);
    R7945(b,6)=rot(1,2);
    R7945(b,7)=rot(2,2);
    R7945(b,8)=rot(3,2);
    R7945(b,9)=rot(1,3);
    R7945(b,10)=rot(2,3);
    R7945(b,11)=rot(3,3);
    yaw7945(b,1) = atan(rot(2,1)/rot(1,1));
    pitch7945(b,1) = atan(-rot(3,1)/sqrt((rot(3,2))^2+(rot(3,3))^2));
    roll7945(b,1) = atan(rot(3,2)/rot(3,3));
end
%%-------------------------------------------------------------------------
%With this part of the script it is stated that if a grain from before movement does not have a match,
%it gets zeros on the place where in other cases the values for the rotation matrix are placed.
%This is useful when a filter for certain set of grains needs to be created in Excel.
%When the yaw/pitch/roll can be placed in line with the other shape parameters of the corresponding grain
%in its initial state, these parameters can easily be selected with the use of the VBA script.
VenE = zeros(x(1,1),14);
for g = 1 : x(1,1)
    for h = 1:p(1,1)
        if R7945(h,1) == K(g,1)
            VenE(g,1) = K(g,1);
            VenE(g,2) = R7945(h,2);
            VenE(g,3) = R7945(h,3);
            VenE(g,4) = R7945(h,4);
            VenE(g,5) = R7945(h,5);
            VenE(g,6) = R7945(h,6);
            VenE(g,7) = R7945(h,7);
            VenE(g,8) = R7945(h,8);
            VenE(g,9) = R7945(h,9);
            VenE(g,10) = R7945(h,10);
            VenE(g,11) = R7945(h,11);
            VenE(g,12) = yaw7945(h,1);
            VenE(g,13) = pitch7945(h,1);
            VenE(g,14) = roll7945(h,1);
            break;
        else
            if h == p(1,1)
            VenE(g,1) = K(g,1);
            VenE(g,2) = 0; VenE(g,3) = 0; VenE(g,4) = 0; VenE(g,5) = 0; VenE(g,6) = 0; VenE(g,7) = 0; VenE(g,8) = 0;
            VenE(g,9) = 0; VenE(g,10) = 0; VenE(g,11) = 0; VenE(g,12) = 0; VenE(g,13) = 0; VenE(g,14) = 0;
            else
            end
        end
    end
end
```

Comparegrains_allparameters.m

This script compares grains from before and after the rotation and also calculates the difference in all shape parameters. Finally it also removes all zeros from the matrix to obtain only the grain numbers for which a result is found.

```
%%To compare and discard grains and calculate movement
%For both matrix A and B the rows consist of the following 9 items:

% [ (excel index) (Area3D) (Vol3D) (BaryCenterX) (BaryCenterY)
% (BaryCenterZ) (OrientationPhi) (OrientationTheta) (Shapa_VA3D)]

%function C = comparegrains_allparameters_final(A,B)
[m] = size(A);
[n] = size(B);
C = zeros(m(1,1),10);
ResMat = zeros(101,10);
y = 1;
for i = 1:m(1,1)
    for j = 1:101
           k=round((i/m(1,1))*n(1,1))+j-52; %In the first place to compensate for the size difference of matrix A and B
and secondly to compute a fluctuation around the given value of j.
        if and(k <= n(1,1), k > 0) %determination of matrix borders
            if B(k,2) > 0 % <== This command does no longer serve purpose but it does not do any harm either. it used
to mean: discard if Area3d=0; when this grain has been used before.
                if B(i,1) < 100000
                    if abs(A(i,3) - B(k,3)) <= -0.000006*(B(i,1)) + 0.001 %This formula and filter for the diff. in
Vol3D has been found by drawing a trendline through the stdev of 40 grains from each 100.
                        ResMat(y,1) = A(i,1);
                        ResMat(y,2) = B(k,1);
                        ResMat(y,3) = A(i,2)-B(k,2); %diff in Area3d
                        ResMat(y,4) = A(i,3)-B(k,3); %diff in Vol3d
                        ResMat(y,5) = A(i,4)-B(k,4); %diff in BaryX
                        ResMat(y,6) = A(i,5)-B(k,5); %diff in BaryY
                        ResMat(y,7) = A(i,6)-B(k,6); %diff in BaryZ
                        ResMat(y,8) = A(i,7)-B(k,7); %diff in OrthPhi
                        ResMat(y,9) = A(i,8)-B(k,8); %diff in OrthTheta
                        ResMat(y,10) = A(i,9)-B(k,9); %diff in Shape_VA3d
                        y = y+1;
                        if j == 101
                            if y == 1 % Here it is determined wheter there are any result and if so, the best result is
determined.
                                C(i,1) = A(i,1);
                                C(i,2)=0;C(i,3)=0;C(i,4)=0;C(i,5)=0;C(i,6)=0;C(i,7)=0;C(i,8)=0;C(i,9)=0;C(i,10)=0;
                            else
                                T = ResMat(:,3);
                                [x] = find(T);
                                [y,z] = size(x);
                                Z = ones(y,10);
                                for b = 1:y
                                    Z(b,:) = ResMat(x(b,1),:);
                                end
                                [Y,I] = min(abs(Z(:,3)));
                                C(i,:) = Z(I,:);
                                y = 1;
                                ResMat = zeros(101,10);
                                break;
                            end
                        else
                        end
                    else
                        if j == 101
                            if y == 1 % Here it is determined wheter there are any result and if so, the best result is
determined.
                                C(i,1) = A(i,1);
                                C(i,2)=0;C(i,3)=0;C(i,4)=0;C(i,5)=0;C(i,6)=0;C(i,7)=0;C(i,8)=0;C(i,9)=0;C(i,10)=0;
                            else
                                T = ResMat(:,3);
                                [x] = find(T);
                                [y,z] = size(x);
                                Z = ones(y,10);
                                for b = 1:y
                                    Z(b,:) = ResMat(x(b,1),:);
                                end
                                [Y,I] = min(abs(Z(:,3)));
                                C(i,:) = Z(I,:);
                                y = 1;
                                ResMat = zeros(101,10);
                                break;
                            end
                        else
                        end
                    end
                else
```

```
                    if abs(A(i,3)- B(k,3)) <=   0 %As mentioned in my report, I have played with the possibility to
apply multiple thresholds. I haven’t used this for my last data processing as this part is only used for B(i,1) >100000
                        ResMat(y,1) = A(i,1);
                        ResMat(y,2) = B(k,1);
                        ResMat(y,3) = A(i,2)-B(k,2); %diff in Area3d
                        ResMat(y,4) = A(i,3)-B(k,3); %diff in Vol3d
                        ResMat(y,5) = A(i,4)-B(k,4); %diff in BaryX
                        ResMat(y,6) = A(i,5)-B(k,5); %diff in BaryY
                        ResMat(y,7) = A(i,6)-B(k,6); %diff in BaryZ
                        ResMat(y,8) = A(i,7)-B(k,7); %diff in OrthPhi
                        ResMat(y,9) = A(i,8)-B(k,8); %diff in OrthTheta
                        ResMat(y,10) = A(i,9)-B(k,9); %diff in Shape_VA3d
                        y = y+1;
                        if j == 101
                            if y == 1 % Here it is determined wheter there are any result and if so, the best result is
determined.
                                C(i,1) = A(i,1);
                                C(i,2)=0;C(i,3)=0;C(i,4)=0;C(i,5)=0;C(i,6)=0;C(i,7)=0;C(i,8)=0;C(i,9)=0;C(i,10)=0;
                            else
                                T = ResMat(:,3);
                                [x] = find(T);
                                [y,z] = size(x);
                                Z = ones(y,10);
                                for b = 1:y
                                    Z(b,:) = ResMat(x(b,1),:);
                                end
                                [Y,I] = min(abs(Z(:,3)));
                                C(i,:) = Z(I,:);
                                y = 1;
                                ResMat = zeros(101,10);
                                break;
                            end
                        else
                        end
                    else
                        if j == 101
                            if y == 1
                                C(i,1) = A(i,1);
                                C(i,2)=0;C(i,3)=0;C(i,4)=0;C(i,5)=0;C(i,6)=0;C(i,7)=0;C(i,8)=0;C(i,9)=0;C(i,10)=0;
                            else
                                T = ResMat(:,3);
                                [x] = find(T);
                        [y,z] = size(x);
                        Z = ones(y,10);
                        for b = 1:y
                            Z(b,:) = ResMat(x(b,1),:);
                        end
                        [Y,I] = min(abs(Z(:,3)));
                        C(i,:) = Z(I,:);
                        y = 1;
                        ResMat = zeros(101,10);
                        break;
                    end
                else
                end
            end
        end
    else
        if j == 101
            if y == 1
                C(i,1) = A(i,1);
                C(i,2)=0;C(i,3)=0;C(i,4)=0;C(i,5)=0;C(i,6)=0;C(i,7)=0;C(i,8)=0;C(i,9)=0;C(i,10)=0;
            else
                T = ResMat(:,3);
                [x] = find(T);
                [y,z] = size(x);
                Z = ones(y,10);
                for b = 1:y
                    Z(b,:) = ResMat(x(b,1),:);
                end
                [Y,I] = min(abs(Z(:,3)));
                C(i,:) = Z(I,:);
                y = 1;
                ResMat = zeros(101,10);
                break;
            end
        else
        end
    end
else
    if j == 101
        if y == 1
            C(i,1) = A(i,1);
            C(i,2)=0;C(i,3)=0;C(i,4)=0;C(i,5)=0;C(i,6)=0;C(i,7)=0;C(i,8)=0;C(i,9)=0;C(i,10)=0;
```

```
                    %If there is no row in matrix B that can be compared
                    %with matrix A, the cow in the output matrix C becomes 0
                else
                    T = ResMat(:,3);
                    [x] = find(T);
                    [y,z] = size(x);
                    Z = ones(y,10);
                    for b = 1:y
                        Z(b,:) = ResMat(x(b,1),:);
                    end
                    [Y,I] = min(abs(Z(:,3)));
                    C(i,:) = Z(I,:);
                    y = 1;
                    ResMat = zeros(101,10);
                    break;
                end
            else
            end
        end
        if k >= n(1,1)
            if y == 1
                C(i,1) = A(i,1);
                C(i,2)=0;C(i,3)=0;C(i,4)=0;C(i,5)=0;C(i,6)=0;C(i,7)=0;C(i,8)=0;C(i,9)=0;C(i,10)=0;
            else
                T = ResMat(:,3);
                [x] = find(T);
                [y,z] = size(x);
                Z = ones(y,10);
                for b = 1:y
                    Z(b,:) = ResMat(x(b,1),:);
                end
             [Y,I] = min(abs(Z(:,3)));
                C(i,:) = Z(I,:);
                y = 1;
                ResMat = zeros(101,10);
                break;
            end
        else
        end
    end
end
%%The following algoritm checks whether there has been a 'double
```

```
%%comparison with a grain of the stirred matrix (the B matrix). The
%%parameters of these results are stored in the "same"-matrix and all but
%%the best result (based on the difference in Vol3D) is rejected. This way
%%we can not have double values in the 2nd column of the C-matrix.
f=2;
count=0;
same = zeros(20,10);
for u = 1:m(1,1) %With this we check if there has been comparison with the same grain from the 'stirred' data set
    if C(u,2)>= 1 % For the grains that have 0 outcomes, nothing needs to be compared.
        same(1,:) = C(u,:);
        for g = 1:201 %The same grain can, for example, be compared by grain 1 and grain 41 from the 'unstirred' data
set (range = +- 40)
            G = u + g - 102;
            if G > 0 %G must be a positive value that lies within matrix C
                if G <= m(1,1) %Whenever G becomes bigger than the size of matrix C, the rest of the outcomes still
needs to be processed. (otherwise these leftover-values remain struck in the 'same'-matrix)
                    if C(G,2) == same(1,2) %If two or more values have been found in column 2 of matrix C that
correspond,than the parameters of this row needs to be put in the research-matrix 'same'.
                        same(f,:) = C(G,:);
                        f = f+1;
                        if g == 201 %When all grains within range have been researched, it needs to be determined which
of these comparisons is valid.
                            T = same(:,1);
                            [x] = find(T);
                            [y,z] = size(x);
                            V = ones(y,10);
                            for b = 1:y
                                V(b,:) = same(x(b,1),:);
                            end
                            [Y,I] = min(abs(V(:,4))); % The result with the smallest difference in Vol3D.
                            for U = 1:m(1,1)
                                if C(U,1) == V(I,1) %The row of the result with the smallest d{Vol3D} in matrix C
remains the same.
                                else
                                    if and(C(U,1) ~= V(I,1), C(U,2) == V(I,2)) %All other comparisons with the same
grain proved to be inferior and need to be rejected. ==> the C matrix becomes 0 for these grains.

C(U,2)=0;C(U,3)=0;C(U,4)=0;C(U,5)=0;C(U,6)=0;C(U,7)=0;C(U,8)=0;C(U,9)=0;C(U,10)=0;
                                    else
                                    end
                                end
                            end
                            f = 2;
                            same = zeros(20,10);
                            count=count+1;
                        else
                        end
                    else
                        if g == 201
                            T = same(:,1);
                            [x] = find(T);
                            [y,z] = size(x);
                            V = ones(y,10);
                            for b = 1:y
                                V(b,:) = same(x(b,1),:);
                            end
                            [Y,I] = min(abs(V(:,4)));
                            for U = 1:m(1,1)
                                if C(U,1) == V(I,1)
                                else
                                    if and(C(U,1) ~= V(I,1), C(U,2) == V(I,2))

C(U,2)=0;C(U,3)=0;C(U,4)=0;C(U,5)=0;C(U,6)=0;C(U,7)=0;C(U,8)=0;C(U,9)=0;C(U,10)=0;
                                    else
                                    end
                                end
                            end
                            f = 2;
                            same = zeros(20,10);
                            count=count+1;
                        else
                        end
                    end
                else
                    T = same(:,1);
                    [x] = find(T);
                    [y,z] = size(x);
                    V = ones(y,10);
                    for b = 1:y
                        V(b,:) = same(x(b,1),:);
                    end
                    [Y,I] = min(abs(V(:,4)));
                    for U = 1:m(1,1)
                        if C(U,1) == V(I,1)
                        else
                            if and(C(U,1) ~= V(I,1), C(U,2) == V(I,2))
                                C(U,2)=0;C(U,3)=0;C(U,4)=0;C(U,5)=0;C(U,6)=0;C(U,7)=0;C(U,8)=0;C(U,9)=0;C(U,10)=0;
                            else
                            end
                        end
                    end
                    f = 2;
                    same = zeros(20,10);
                    count=count+1;
                end
            else
            end
        end
    else
    end
end

%%now starts the script to that removes the zeros from the matrix
Q = C(:,2);
[j] = find(Q);
[p,q] = size(j);
S = ones(p,10);
for k = 1:p
     S(k,:) = C(j(k,1),:);
end
```

Appendix 3: VBA script

With the use of this script it is possible to select the rows where the values of a chosen column fall in a certain range.

```
Option Explicit

Sub SelectByValue(Rng1 As Range, MinimunValue As Double, MaximumValue As Double)

    Dim MyRange As Range
    Dim Cell As Object

     'Check every cell in the range for matching criteria.
    For Each Cell In Rng1
        If Cell.Value >= MinimunValue And Cell.Value <= MaximumValue Then
            If MyRange Is Nothing Then
                Set MyRange = Range(Cell.Address)
            Else
                Set MyRange = Union(MyRange.EntireRow, Range(Cell.Address))
            End If
        End If
    Next
     'Select the new range of only matching criteria
         MyRange.EntireRow.Select

End Sub

Sub CallSelectByValue()

     'Call the macro and pass all the required variables to it.
     'In the line below, change the Range, Minimum Value, and Maximum Value as needed
    Call SelectByValue(Range("U3:U1715"), 1, 2)

End Sub
```