FRACTURE DETECTION AND PROPERTY ESTIMATION IN MICROCT IMAGES

Omar Alamoudi (supervised by Dr. Nicola Tisato)

*Department of Geological Sciences The University of Texas at Austin*

ABSTRACT

I need to add abstract

1 INTRODUCTION

In applications that utilize and depend on the understanding of multi-phase composite material properties, understanding the effects of apparent properties is generally a good approximation of how the composite behaves. It is also important to understand how the variation of the composite constituents affect said apparent properties. The behavior of such composites varies under the influence of different physical and chemical processes. For example, a primary feature that helps in understanding how composites constituents affect one another pertains to their interaction along boundaries of contact. Rocks, soils, and rock melts are three composite material examples that geologist, geophysicists, and engineers want to understand. Rocks are composed of a multiple material phases. A rock sample is composed of at least two material phases: a solid, and a fluid phase. Considering a rock sample with an economic fluid as its fluid phase, the interaction between that fluid and the rock solid phase is governed by the physical properties of each phase, the conservation of mass, momentum, and energy of their system; and the geometric distribution of these phases in the rock sample. Also, subject the sample to a deformation field, causes the geometric distribution to change, and contributes to the balance laws mentioned. With this broad view points in mind, a particular physical process that we are interested in is the processes governing fluid flow within fractured rocks under a deformation regime. The question of how fluid flow is affected by the induction and evolution of fractures in such rocks is of interest.

Since fluid flow is generally dominated by flow through fracture (Zoback, 2007), fracture geometric properties are primary influences on the fluid flow. An intuitive property to consider here is the fracture aperture (Zimmerman and Bodvarsson, 1996). Other properties such as the fracture volume, and fracture surface area are also important but are more difficult to measure (Zhao et al., 2020). Also, the evolution of these fractures due to deformation is relevant to understanding the change in fluid flow in a fractured sample. In order to understand how these properties affect the fluid flow, we need to detect and measure these fracture properties through time.

In order to understand how the evolution of fractures influences samples permeability, we plan to conduct tri-axial testing of cylindrical rock samples while simultaneously acquiring micrometerscale Computed Tomography images (microCT), permeability measurements, and ultrasonic P and S wave velocities at multiple time points. A time-sequential acquisitions of these measurements will provide the means to capture the effects of fracture induction and evolution on fluid permeability. The acquisition of all types of data will be conducted at different deformation stages.

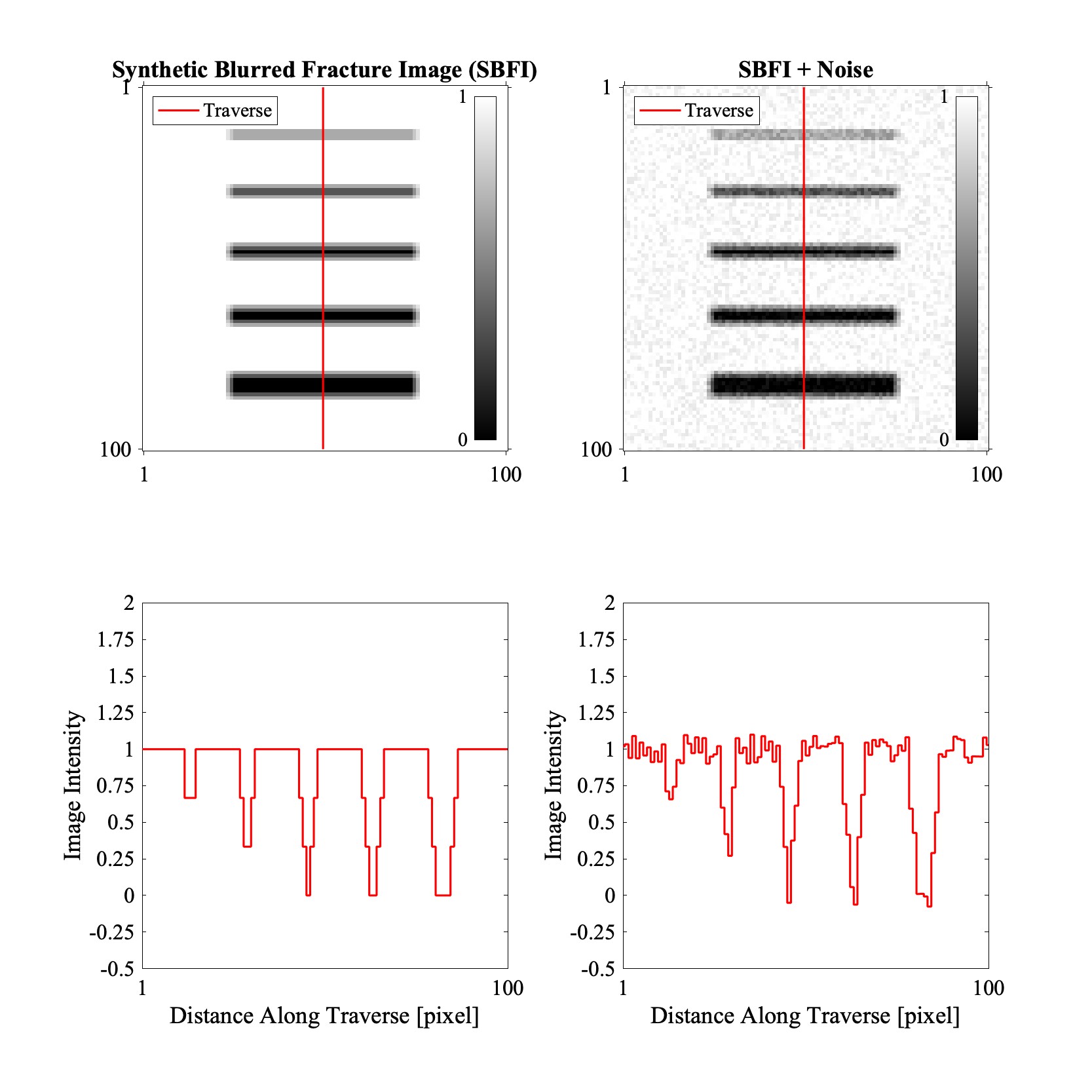
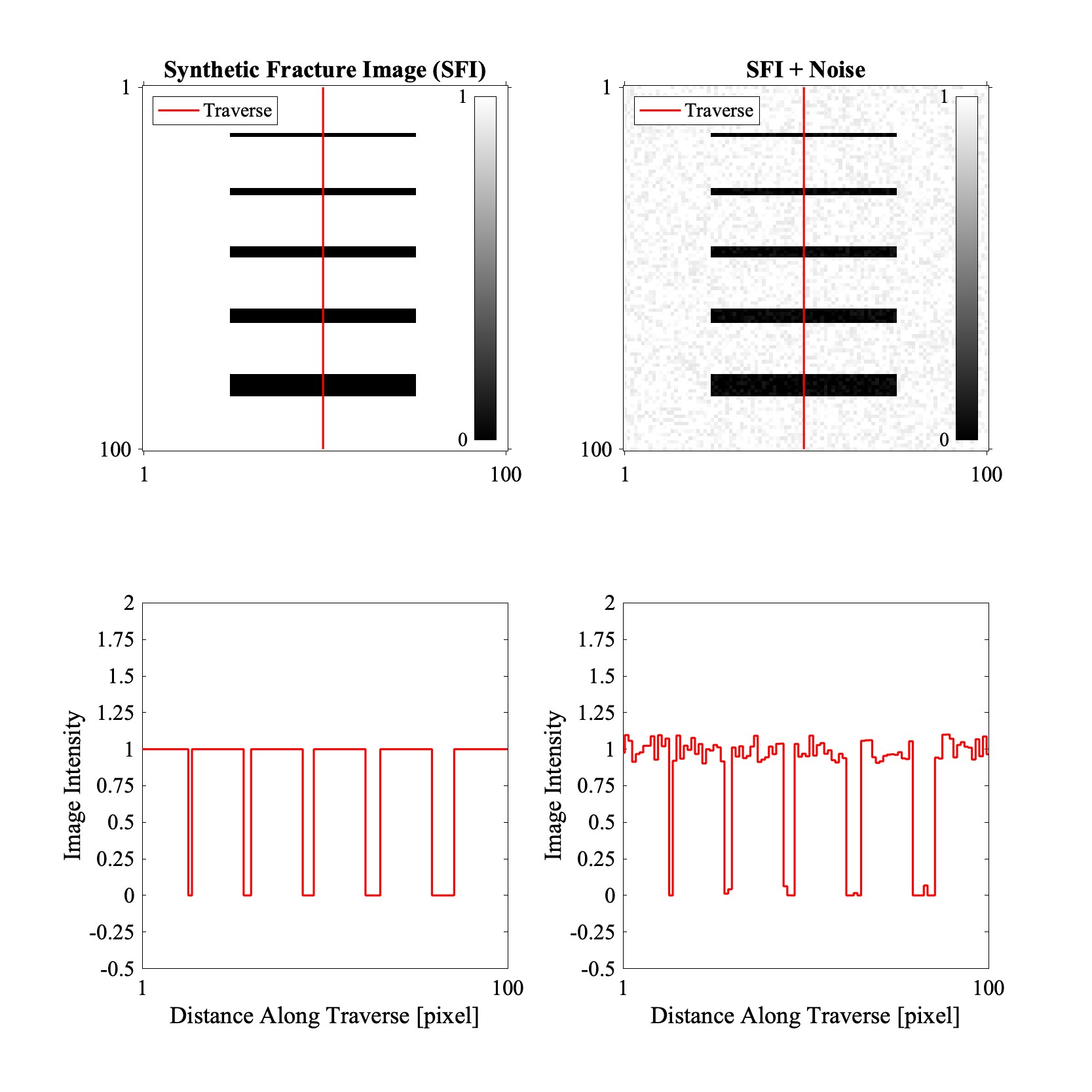
Therefore, we need a systematic, semi-automatic approach to reliably and consistently detect and measure the fractures of the samples. In this paper we focus on detecting fractures and measuring their surface area.

1. MATERIALS AND METHODS

Our main objective is to improve the ability to detect, and henceforth measure fractures in microCT images acquired of rock samples in a systematic, semi-automatic, repeatable way. The necessity of semi-automatically detecting and measuring fractures is due to our over arching goal to use the developed methodology to measure the evolving fracture properties in microCT images over time of a sample undergoing deformation and fracturing.

The methodology used is based on the one described in Voorn et al. (2013). To demonstrate this approach in detecting fractures, we first apply the methodology on 2D and 3D synthetic images that we generate resembling fractures. We then compute the spatial Hessians for each voxel of the synthetic images, and finally conduct further analysis to map the fracture features from the images. The description of these steps follows in the same order stated.

* 1. Synthetic Image Generation



(a) Set of ”sharp” images (b) Set of blurry images

Figure 1. The middle slice of a three-dimensional synthetic images. Top row shows the images, and the bottom row shows a traverse along the vertical red line.

The workflow used in generating the synthetic fracture images was done using Matlab. The objective of creating these images is to calibrate the outcome of analysis that will be discussed in later in section (2.3) and section (2.4). A function was created that takes four input parameters and generates a synthetic image of a white background and black rectangles, for two-dimensional images, and cuboids for three-dimensional images. The input parameters are: (1) the length a long the x and y axis of the square synthetic image; (2) an array of integers representing the fracture apertures in pixels/voxels, i.e. [1,2,3,4,6]; (3) a boolean flag specifying if random noise should be added to the image; (4) a signal to noise (SNR). For every function call, an equally spaced vertical fractures are drawn into an image as pixels with the value zero in a background of value one. The resultant image is a square image with an edge length specified by the first input argument see figure (1).

In order to understand the effects of blurring and noise in our analysis, four different images are generated using identical image size and fracture apertures in two sets. The first set is of two ”sharp” images, and the other two are blurred images. The blurring is produced by the convolution of the sharp image with a 3-by-3 (-by-3 in 3D) box filter. The blurred images are a rudimentary attempt to represent the effect of the Point Spread Function (PSF) of a microCT machine (Ketcham et al., 2010). In each set, one image is noise free while random noise was added to the other. This is our attempt to observe and quantify the effect of random noise. The result of this workflow is a two-dimensional image with the specified fracture apertures as the apparent fractures, but since our objective is to map fractures, a plate like geometrical features, Voorn et al. (2013) have stated that two-dimensional images are insufficient in mapping fractures. To generate a three-dimensional image with fractures that are vertically oriented, a stack of three identical images are superimposed on one another. For the images that contain noise, three different realizations of noise are stacked to preserve the randomness of noise. Figure (1) shows the middle slice of a three-dimensional image with input parameters [1,2,3,4,6] for fracture apertures, 100 voxels for the length of the image, and 10 for the SNR.

* 1. MicroCT Images and Sample Description

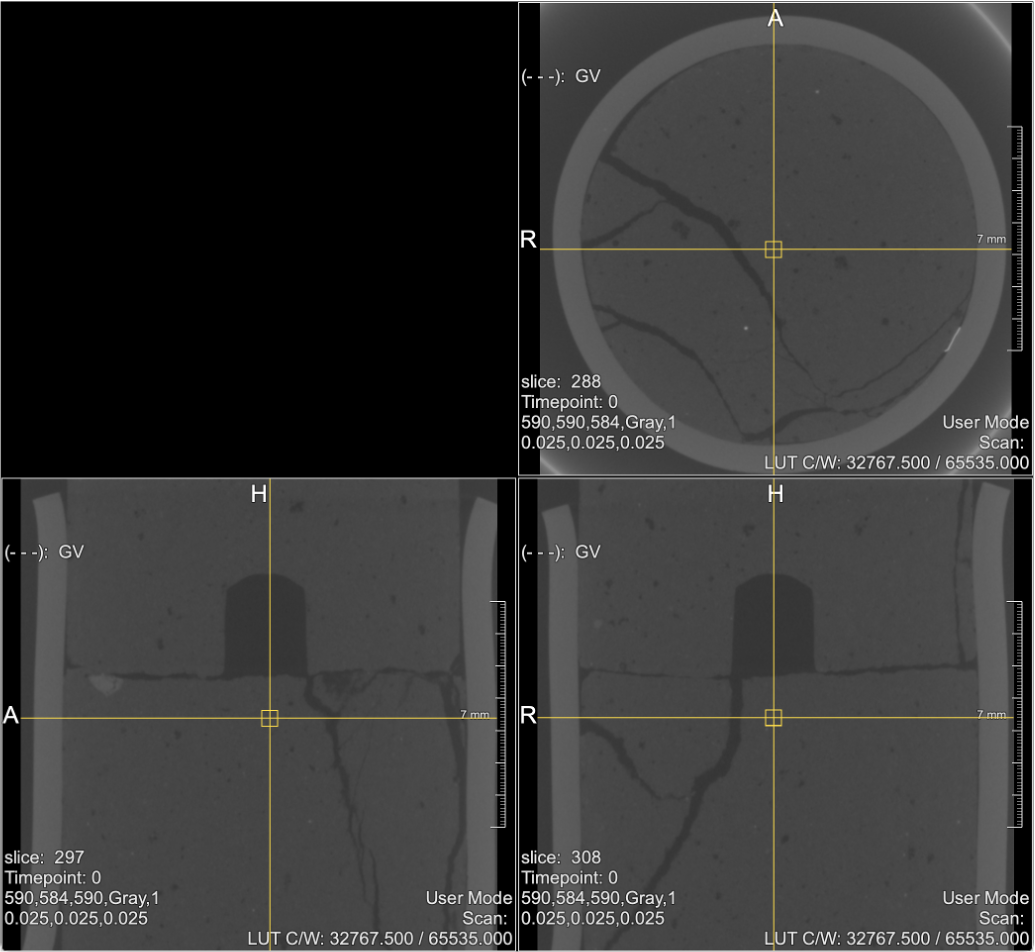


Figure 2. An orthogonal view of the image 18 degree. This view is generated using the ’OrthoView2D’ of MeVisLab software. Both semi-samples are shown. The top semi-sample shows the cavity. Fractures shown are a result of rotating the top semi-sample while applying normal force.

The microCT images used were available through Zhao et al. (2020) and described in Zhao et al. (2017, 2020). In summary, the image volume comprises 1024 slices a long the z axis, and 794 voxels along the x and y axes. Each voxel holds a 16-bit gray value corresponding to approximately the amount of X-ray attenuation incurred by the material in that voxel. The voxels are isotropic cubes with edge length of 25micrometer.

The images are for a cylindrical sample that underwent a rotary shear experiment. Six threedimensional microCT image volumes are available. Here we use two images that were acquired at two different time steps of the experiment. The image from first time step is named 0 degree, and the image from the third time step which is called 18 degree, see figure (2). This provides us the means to observe difference in fracture induction in the sample over time. The scanned sample is a cylindrical flowstone sample that is 12mm in diameter and 32mm in length. After casting and curing, the sample was carefully split horizontally into two cylindrical semi-samples with a rough surface in the middle representing a horizontal fault surface. A semi-conical cavity of 2.5mm diameter and 3mm in depth was drilled into the top semi-sample. This cavity was created to eliminate the potential of the sample fracturing due to compression since the displacement of the central portion of the material is minimal. The sample was intended to fracture only due to rotary shear motion. Since the sample was attached to a stainless steel sample holder, analysis were conducted on part of the sample focused around the horizontal fault. The portion of the images closes to the sample holder was cropped out yielding a total of 620 slices around the horizontal fault see figure (3). The slices used are slices number 137-756.

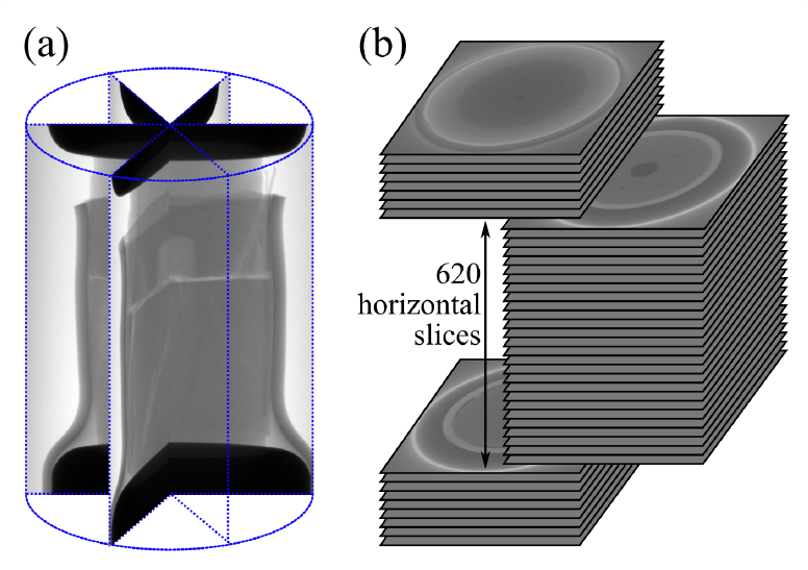


Figure 3. (a) Radial cross sections showing the entire sample. (b) The cropped volume used in the analysis. After Zhao et al. (2017)

2.3 Enhancing Fractures in microCT Images

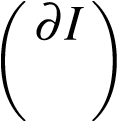
2.3.1 Computing the Hessian

To compute the Hessian of a microCT image, consider a three dimensional Euclidean domain *E*3 representing the domain of a microCT image, and given a scalar valued field *I* : *E*3 → R representing the gray value of every voxel. The Hessian of the scalar field *I* is H(*I*). Consider the matrix notation of the Hessian as shown in eq. (1).

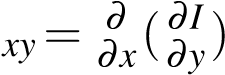
|  |  |
| --- | --- |
|    *I,xx I,xy I,xz*   1. = *I,yx I,yy I,yz*      *I,zx I,zy I,zz*  Where   1. = microCT image gray value field, or image intensity | (1) |

*∂*

*I,ij* =  *. i, j* ∈ {*x,y,z*} (2)

*∂i ∂ j*

The Hessian is a tensor value function H : *E*3 → R3×R3. It has nine components six of which are unique because the Hessian is symmetric. *Hij* denotes the component of the ith row and jth column. The indices *i, j* ∈ {*x,y,z*}. *x,y*, and *z* are right-handed orthogonal coordinates of a defined Euclidean space *E*3. Since microCT images, or digital images in general, are discrete in nature, the computation of the derivatives can be done by finite differences for example. But that is not adequate since fracture features are of different scales. Therefore, a *multi-scale* technique is more useful.

An over view of such technique is provided by Lindeberg (1998). This technique considers different scales of edge features, in our case represented by fractures. For example, the second derivative *I,* of a three-dimensional image represented by a scalar valued function (gray value) *I* = *I*(*x,y,z*) for every voxel with coordinates (*x,y,z*) is computed as the convolution of the image *I* with the second derivative of the three-dimentional Gaussian *G,xy* as following:

*I,* (3)

Where

*α* = Normalization factor

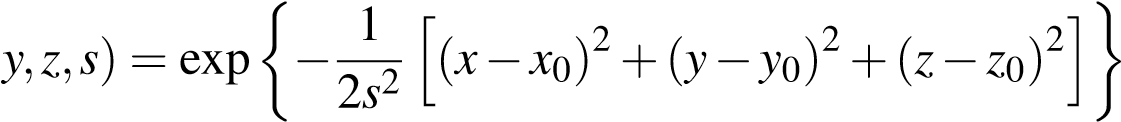
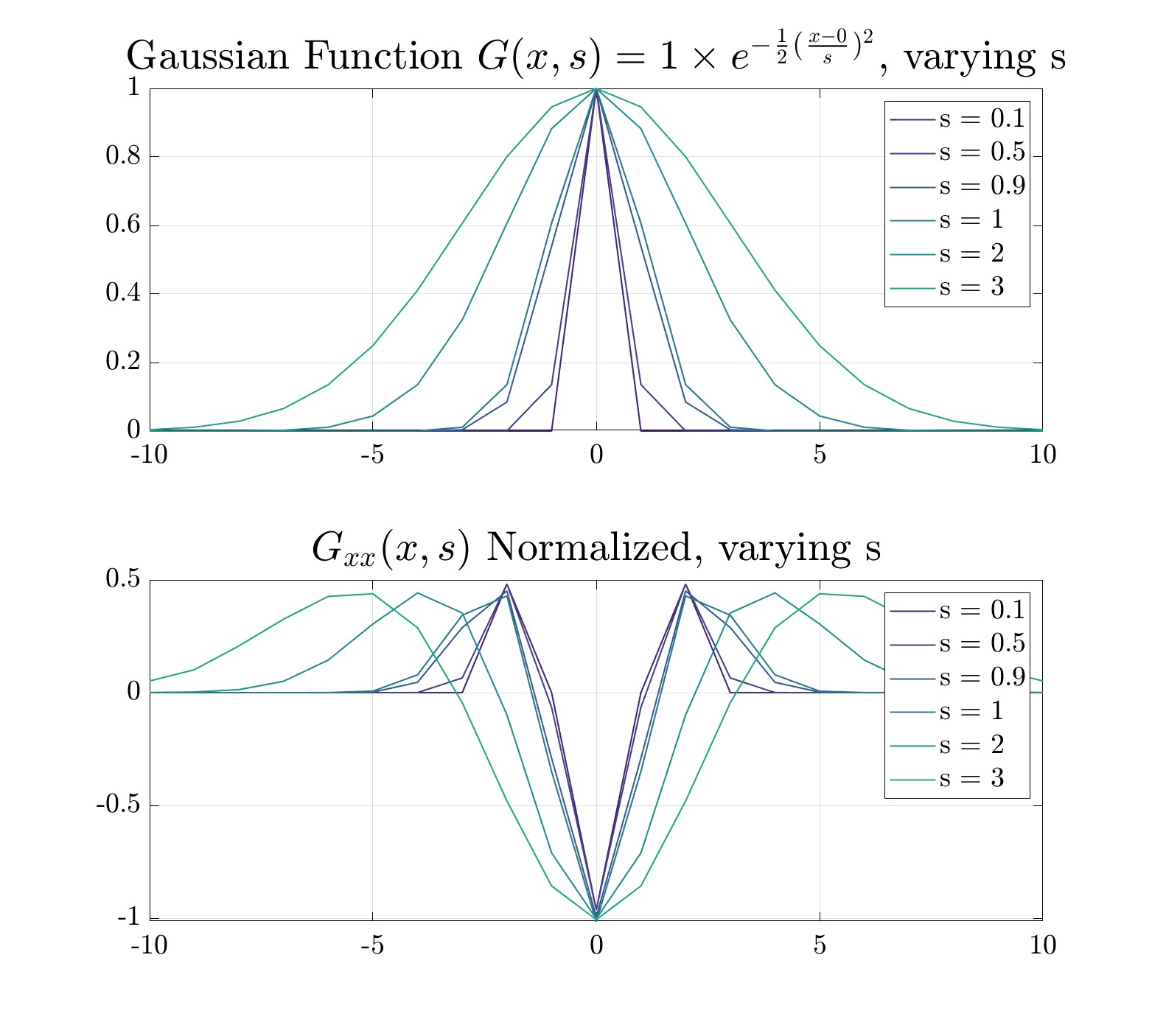
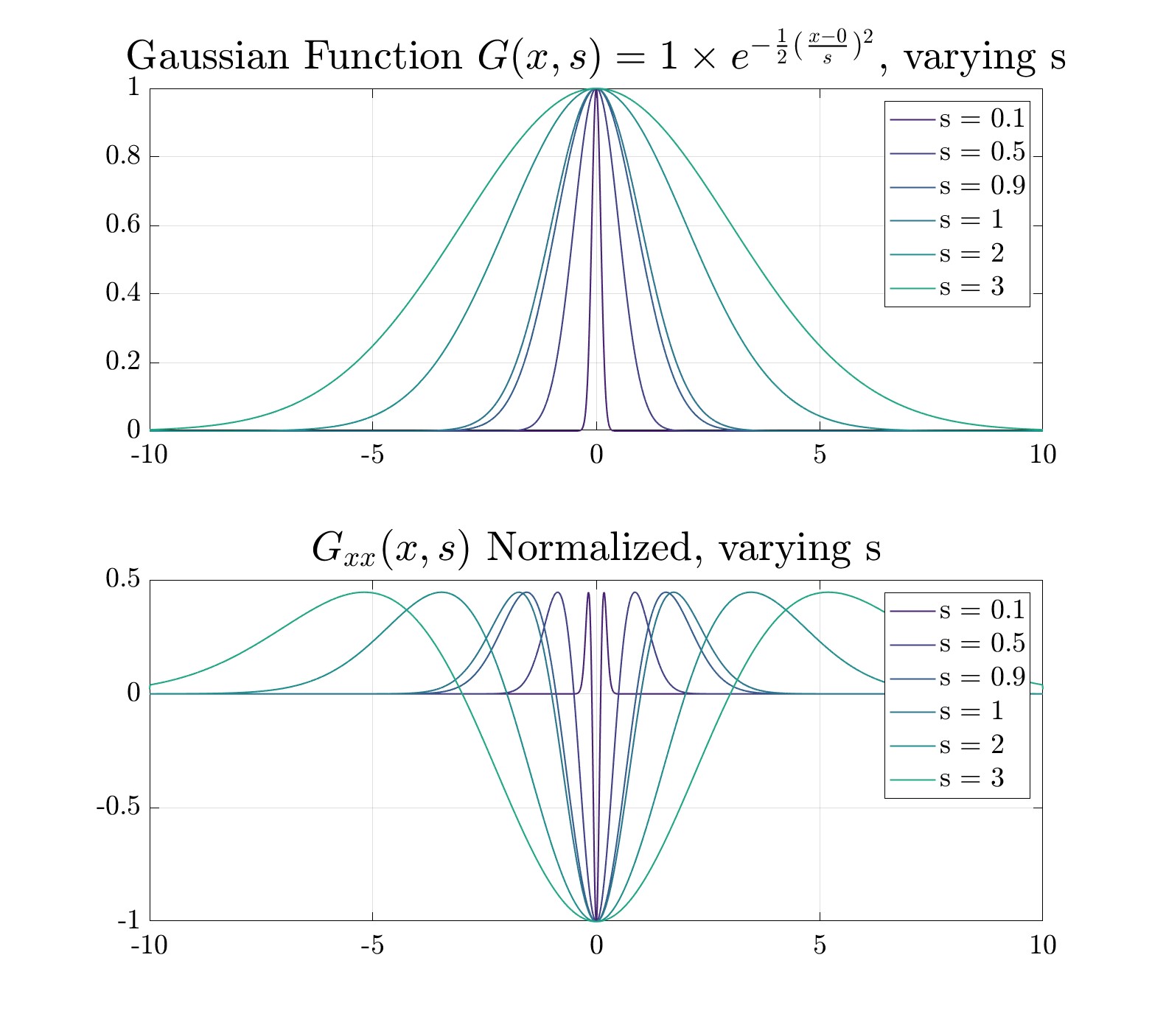
*G*(*x,* (Gaussian)

Figure (4) shows a one-dimensional Gaussian function and the second derivative in a continuous and a discrete form. The second derivative is equivalent to the *G,xx* component of the multi-dimentional Gaussian function. This is the kernal that is convolved with the intensity of the image as shown in eq. (3). It is important to note that all of the operations carried out here is done using the discrete form. Different curves shown in figure (4) correspond to a different scaling parameter *s*. Darker colors correspond to small *s*, while brighter colors correspond to a larger *s*.

For a given image, I computed all nine components of the Hessian for each voxel for different values of the scaling parameter *s*. This was done by (1) choosing a value for *s*; (2) generating the proper three-dimensional Gaussian filter *G*(*x,y,z*) using the MATLAB Image Processing Toolbox



(a) Continuous (b) Discrete

Figure 4. One-dimensional Gaussian function and its second derivative with varying *s* in continuous and discrete forms.

function fspecial3 with a filter size of either 18 or 19. The size of the filter was dependent on 2*s*. The filter was 18 if 2*s* was even and 19 if 2*s* was odd.(3) computing the nine derivatives, *Gji*(*x,y,x*) for *i, j* = [*x,y,z*], that are used in eq. (3), then convolving the result with the images while choosing *α* = 1. Looking at the bottom plot of figure (4a), each curve can be defined by it largest trough bounded by two smaller peaks. The distance that defines the largest trough for every curve between the two points (closest to zero) intersection the horizontal axis is 2*s*.

2.3.2 Fracture Enhancement Analysis

After computing the all nine components of the Hessian for every voxel of the synthetic images for every value of *s*, we compute the eigenvalues *λ*1*, λ*2*, λ*3, where *λ*1 *<λ*2 *<λ*3. I used the Matlab function eig to do so. For every value of *s*, within and close to plate like features, we expect one of the eigenvalues to have a large magnitude while the other two to have small magnitudes. Using the same criteria proposed by Voorn et al. (2013), we can then define a new value *As* for every voxel such that

(

*λ*3−|*λ*2|−|*λ*1| if (*λ*3−|*λ*2|−|*λ*1|) *>* 0

*As* =

1. otherwise

where the subscript *s* here denotes for the scaling parameter. We then normalize *As* such that

*Bs* = max*A*(*sAs*), and finally define another quantity *Cs* such that

(

1. if *Bs >* 1−*γ*

*Cs* = (4)

0 otherwise

*γ* is an arbitrary tolerance that I chose to be 0*.*70 in my application. I then summed up all of *Cs*

for all scaling parameters used. In my case, since this analysis was applied to a synthetic image set, the scaling parameter was set to be *s* . Figure (5) shows the results of conducting the above steps for every value *s*.

The final step is combining all different images for every value of *s*, and inverting the image to match the original synthetic image as shown in figure (6).

2.3.3 Using Voorn et al. (2013) Code

The code described in Voorn et al. (2013) to enhance multi-scale fractures is available in code via a GitHub repository. At first, I recreated the workflow in Matlab with the intention of attempting to further address sub-voxel fractures with minor success not discussed here. More work is needed to achieve this objective. In section (3.1) I will discuss the difference in the outcome of my code compared to the outcome of the code published by Voorn et al. (2013). The synthetic image used in this comparison is the Synthetic Blurry Fracture Image with added noise (SBFIN) that is shown in figure (1b). The results of using my code compared to the published code is shown in figure (6).

2.4 Measuring Fracture Surface area

A particular fracture property of interest is the surface area of the fracture. Fracture surface area is the area of the interface between the ’void’ and the solid phase of a rock sample. The importance of quantifying this geometric feature stems from its influence on the following:

* it is the surface across which fluid exchange between the two phases occur
* the fluid flow mechanisms depends on the orientation of flow with respect to the fracture surface
* the estimation of system energy consumption due to fracturing (surface energy)

For the stated reasons above, we wanted to assess the usage of the Hessian filtering enhanced images compared to a microCT images see figure (7).

2.5 Measuring Surface Area Due to Fracturing

I measured the surface area created by the fracturing process using the isosurface module in MeVisLab. The computation procedure went as following: (1) imported both images volumes (0 degree and 18 degree) into MeVisLab using the module Compose3DFrom2DFiles; (2) set the proper voxel spatial scaling using the module SetWorldMatrix; (3) for regular microCT images, I inverted the image using the Invert(Max-Img) function in Arithmatic1 module; (4) re-scaled where the minimum has the value 0 and the maximum has the value 1; (5) by visually inspecting the resultant images, I identified an inverted gray value that correspond to the same value at a fracture wall and is consistent in both images; (6) computed the surface area using the isosurface module of the sample for both images (0 degree and 18 degree).

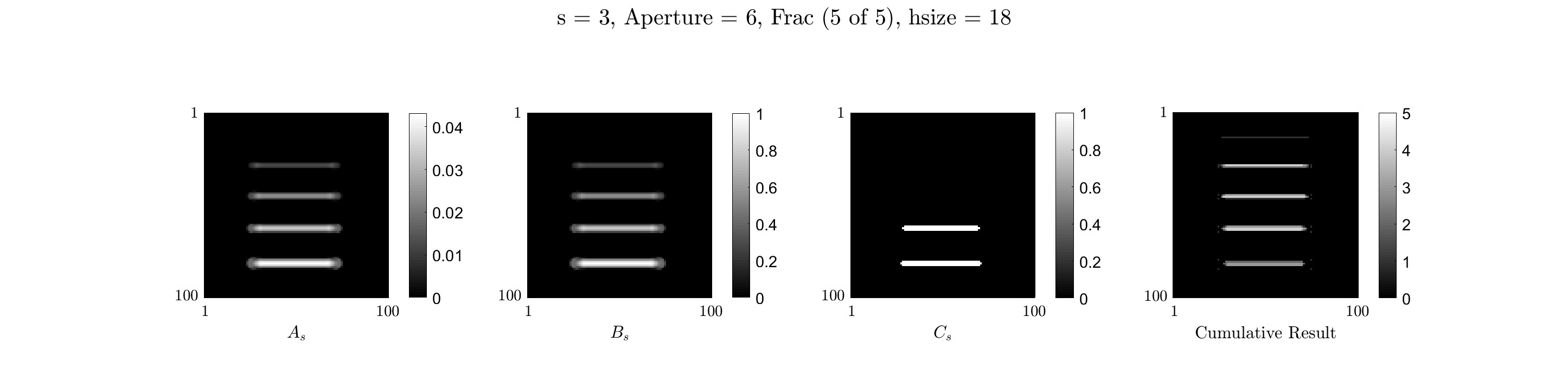
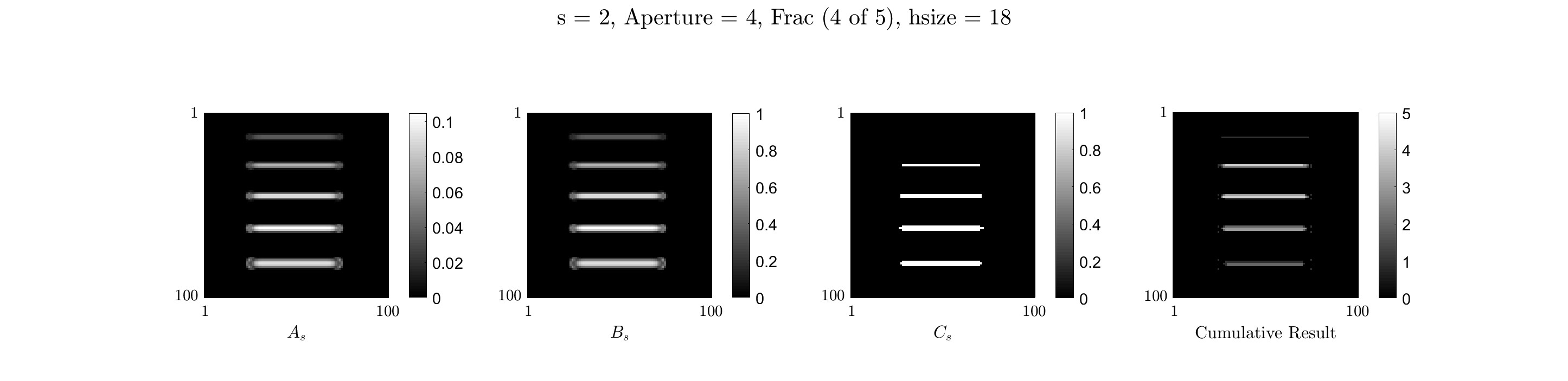
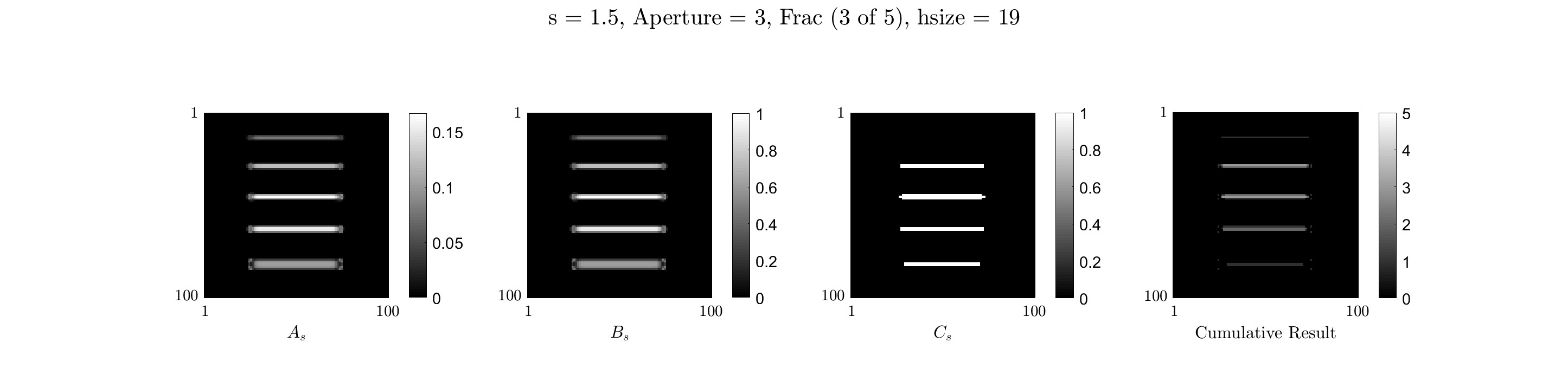
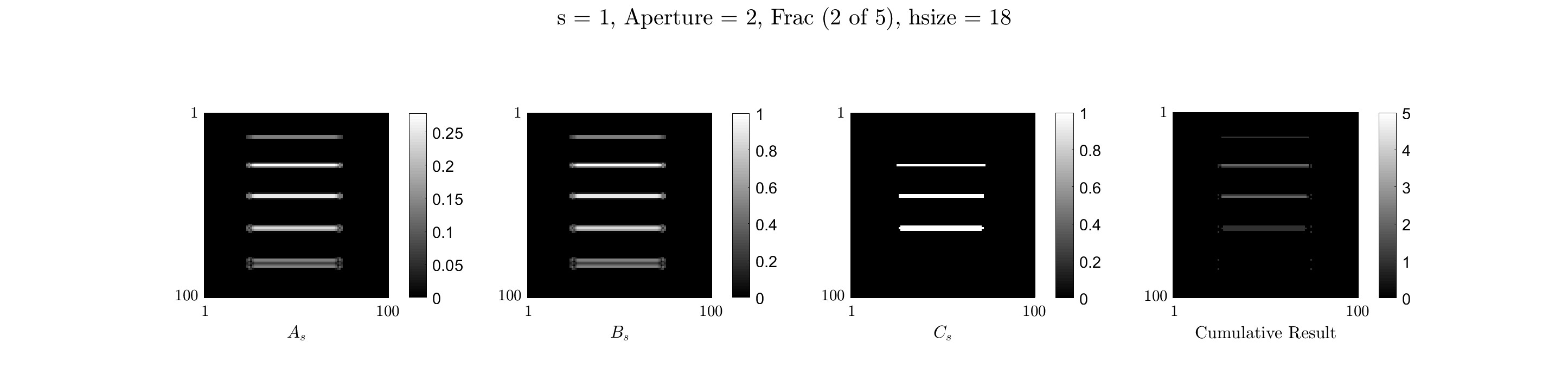
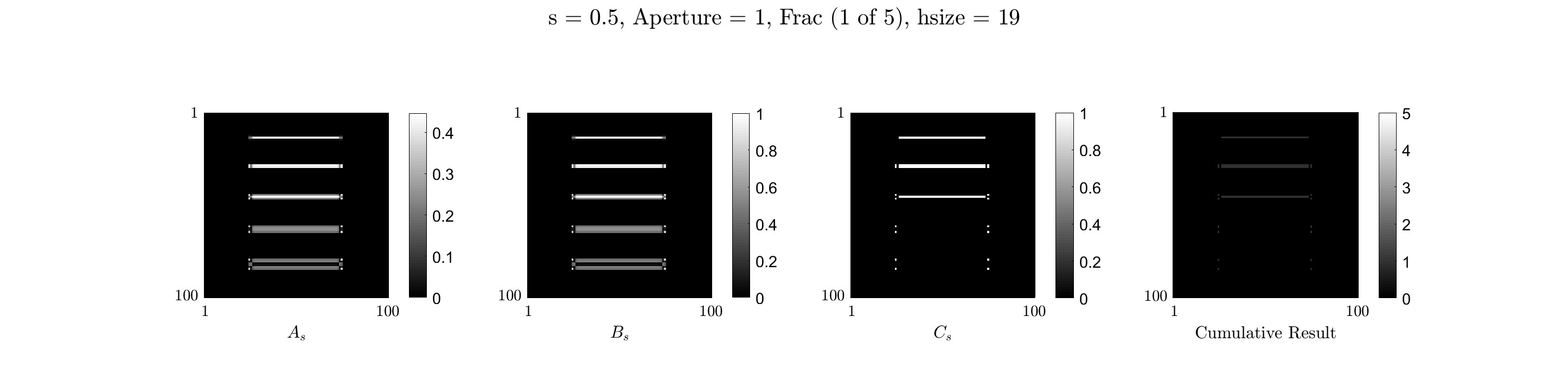


Figure 5. From top to bottom, incremental results for every scaling value *s*. Left to right of each row *As*, *Bs*, *Cs*, and the cumulative addition of the current step and all of the previous ones of *Cs*. Note here that the bottom row shows the final result of the analysis before normalizing and inverting the image.

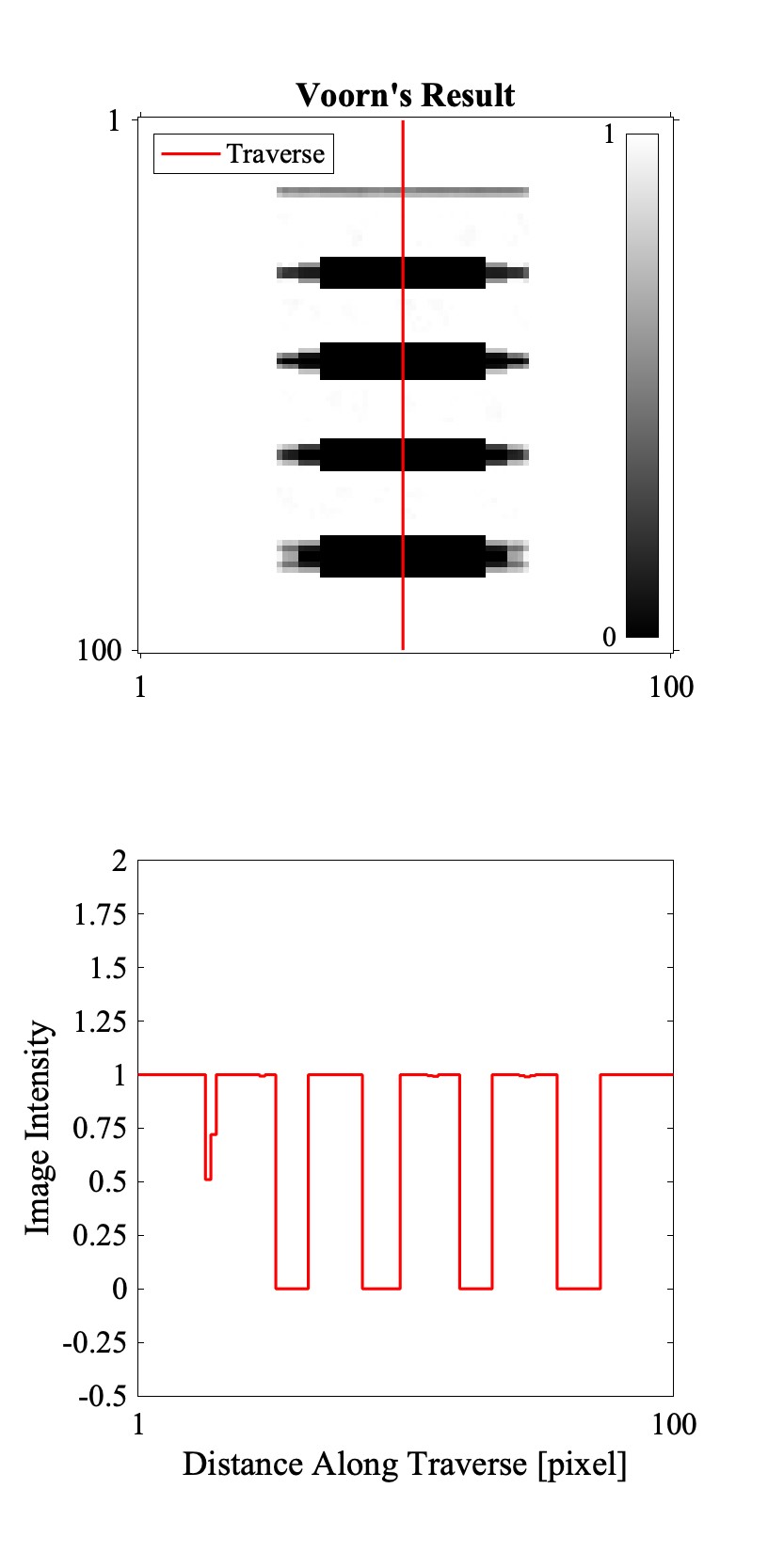
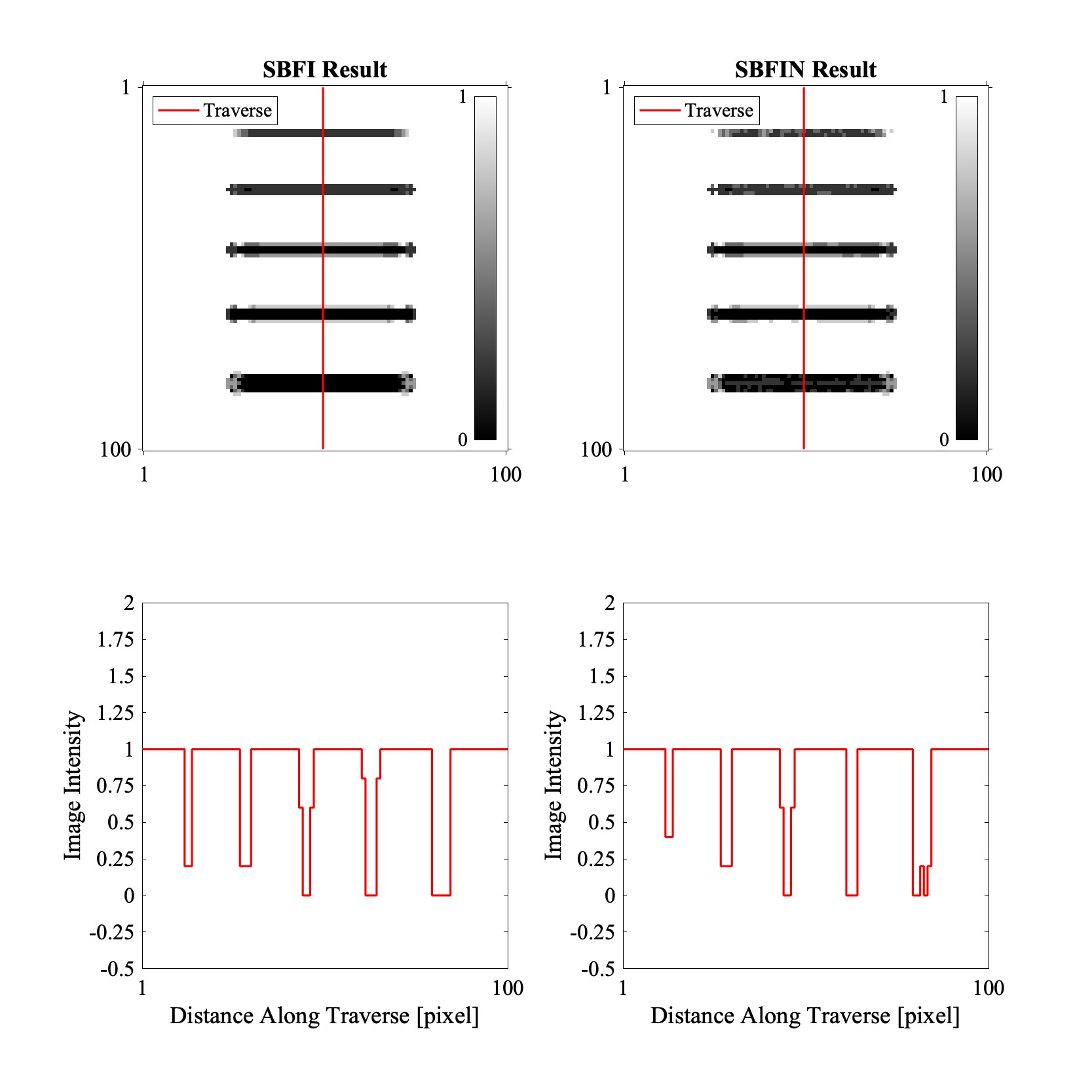


Figure 6. The results of using the Hessian filtering technique on synthetic fracture images. From left to right: result from my implementation on the Synthetic Blurry Fracture Image (SBFI); Result from my implementation on the Synthetic Blurry Fracture Image with Noise; Result from using the published code by Voorn et al. (2013). The bottom row shows the image values along the vertical red line.

1. RESULTS AND DISCUSSION
   1. Comparing the outcomes of my code and Voorn et al. (2013)

Figure (6) shows a significant difference between the results from my implementation compared to the results from Voorn et al. (2013). At this moment, the exact reason why these results vary significantly is unclear. The full extent of the analysis discussed in section (2.3.2) was conducted about three semesters ago. Additional work is need to further investigate the difference in the outcome. However, I can think of a few reasons why there is a significant difference:

* + - the tolerance parameter *γ* stated in eq. (4) is arbitrary
    - filter size used when performing the convolution of the image and the Hessian componenet. In my implementation, I chose the filter size to be 19 voxels if 2*s* is odd, and 18 if 2*s* is even. The reasoning for this is described in section (3.3)

In addition, my implementation resulted in showing the contrast in fractures sizes better than the Voorn et al. (2013), compare the profiles of the gray values in figure (6).

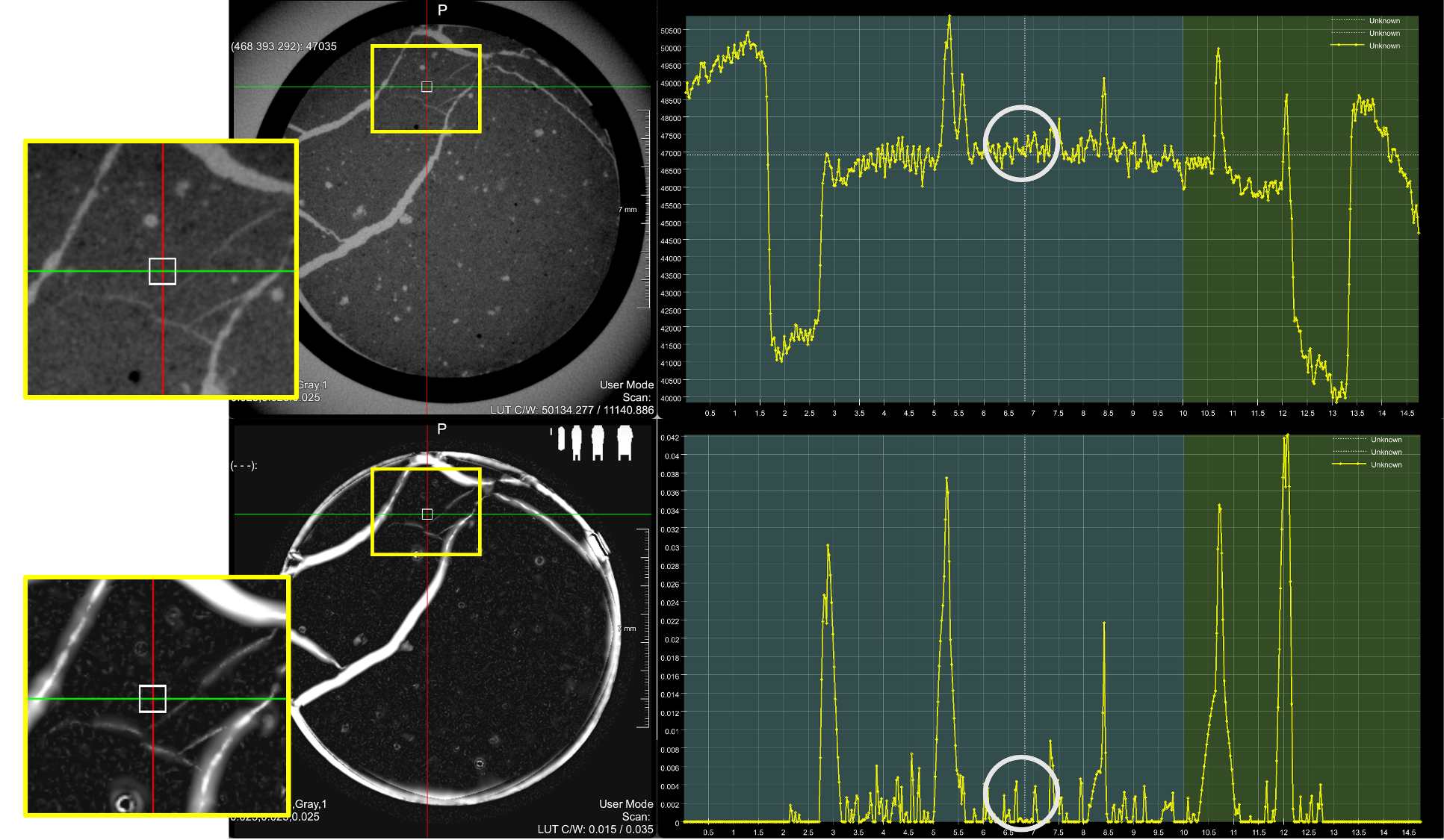


Figure 7. The top row show from left to right: a zoomed in version of the inverted microCT image 18 degree, a slice of the microCT image with two crossed green and red lines and a small while box showing the position of the cursor. The linear plot to the right show the gray values of the inverted microCT image along the green line shown on the microCT slice. Similarly, the bottom row is of the Hessian filtering enhanced image. Notice how much better small fractures show in the enhanced image in the slice image and in the linear plot inside the white circles.

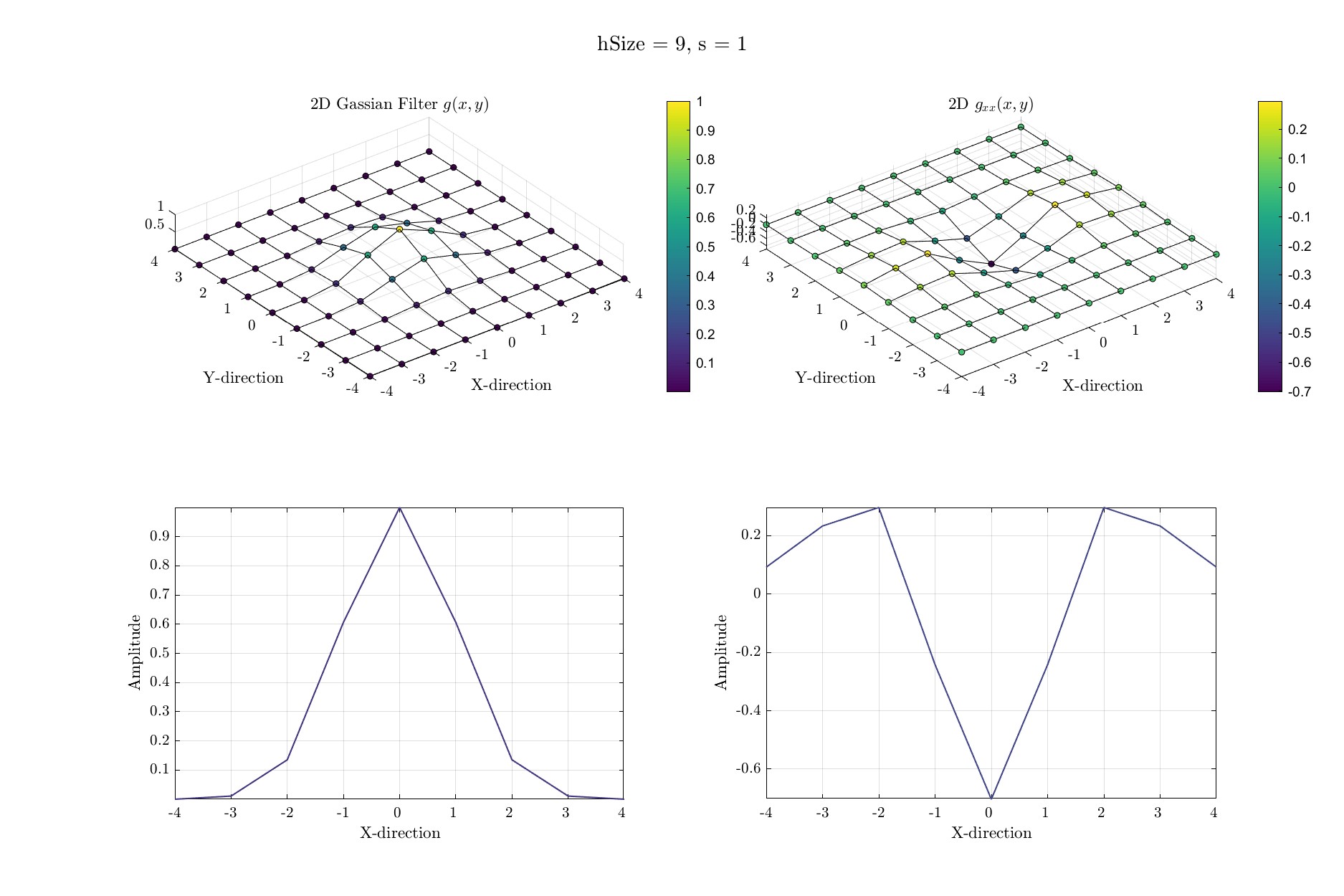
* 1. Surface Area due to Fracturing

The surface area due to fractures is the difference in surface area of these two images 0 degree and 18 degree. Here the surface area was found to be 762mm2. This result is consistent with the published results by (Zhao et al., 2020, figure 4). See the fracture surface area measurements for the average slip distance 1mm in figure (8).

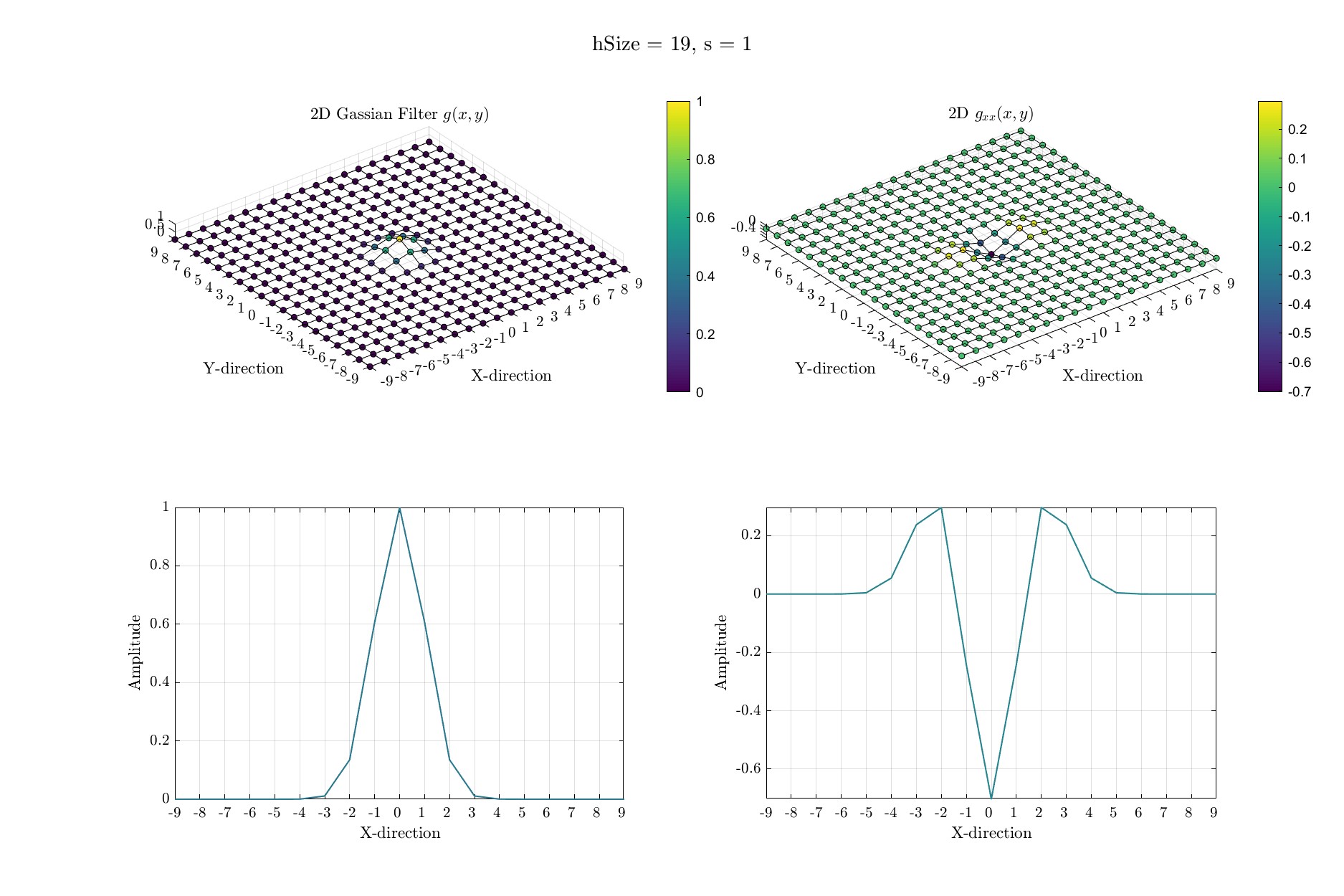
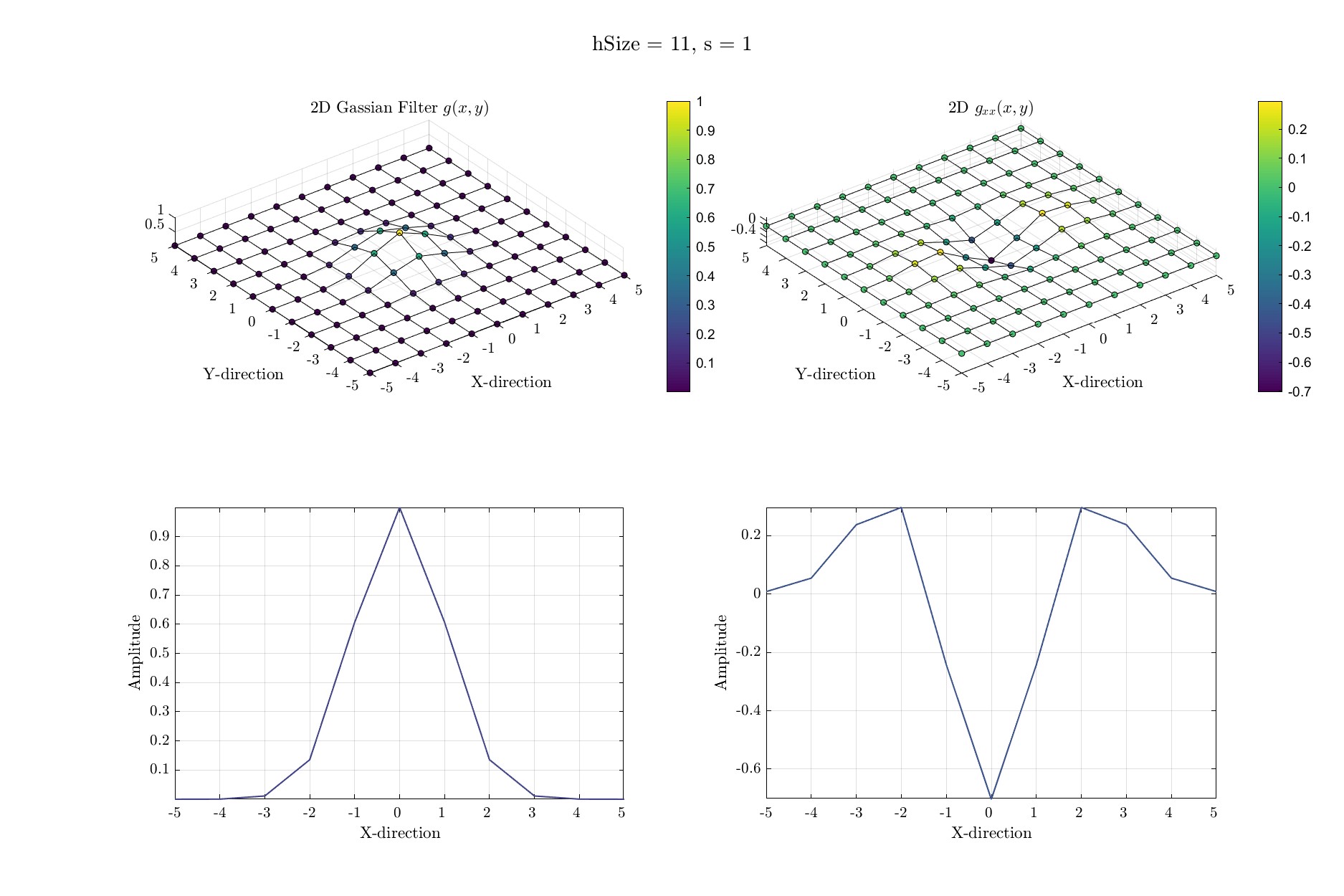
* 1. Further Comment on My Implementation of the Hessian Filtering Technique

Figure (6) shows the resulst of my implementation on the SBFIN images. We can see from the traverse profiles that we are able to detect the fractures of different scales. Although we are able to detect the fractures, not all fracture apertures visually match the corresponding fractures in the original synthetic image in figure (1).

Since our data set and filtering kernels used in the convolution operation are made out of discrete values, the shape of the filter must be preserved. This means that when using the Matlab function fspecial3 to generate the 3D filter, one needs to plot it and insure that the size of the filter results in second derivative filter shape that preserves the two-peak-one-trough shape for a specific scaling parameter *s*. To show this better, consider the 2D filter since it is easier to visualize. Figure (9) shows the kernel generated using the *gaussian* option of the Matlab function fspecial2 (2D equivalent of fspecial3). Shown in this figure are four different filters for one scaling parameter *s* = 1. The filter sizes shown are 5, 9, 11, and 19. Figure (9a) shows that the second derivative *Gxx* shown in the right column does not preserver the trough-peak-trough shape required to produce the appropriate results. But a filter size of 9 or greater seem to preserve that shape, making such sizes sufficient to produce the correct results. The main advantage of keeping the filter size small however, is reducing the computing expense. For my synthetic images, which are 100 x 100 x 3 voxels in size, the computational time was minimal, but considering an actual image of a rock, the computations could become very expensive. An attempt that is not shown here on an image of size 1000 x 350 x 3 was made, and the computational time was about 15 minutes. This shows that there should be an optimal filter size for every scaling parameter *s* to optimize the computational expense.



(a) Filter size = 5 (b) Filter size = 9



(c) Filter size = 11 (d) Filter size = 19

Figure 9. Two-dimensional Gaussian filter *G*(*x,y*) (top left of each set), and *Gxx*(*x,y*) (top right of each set) for *s*=1 and different filter sizes used in generating the kernels by the MATLAB function fspecial3. The bottom row of each set is a traverse line along a line with y-axis value = 0.

1. CONCLUSIONS AND FUTURE WORK

The approach proposed by Voorn et al. (2013) seems to be successful at detecting fractures of multiple scales. There are a few parameters in the algorithm, such as the tolerance parameter *γ* used in computing *Cs* that require some trial and error to optimize the results. Further analysis on larger synthetic images and microCT rock images can be used to optimized the approach further. The size of the filters used in the computation of the Hessian are critical to produce correct results.

Finally, further investigations into optimizing this approach to recover the fracture apertures is of value and will be considered in future work.

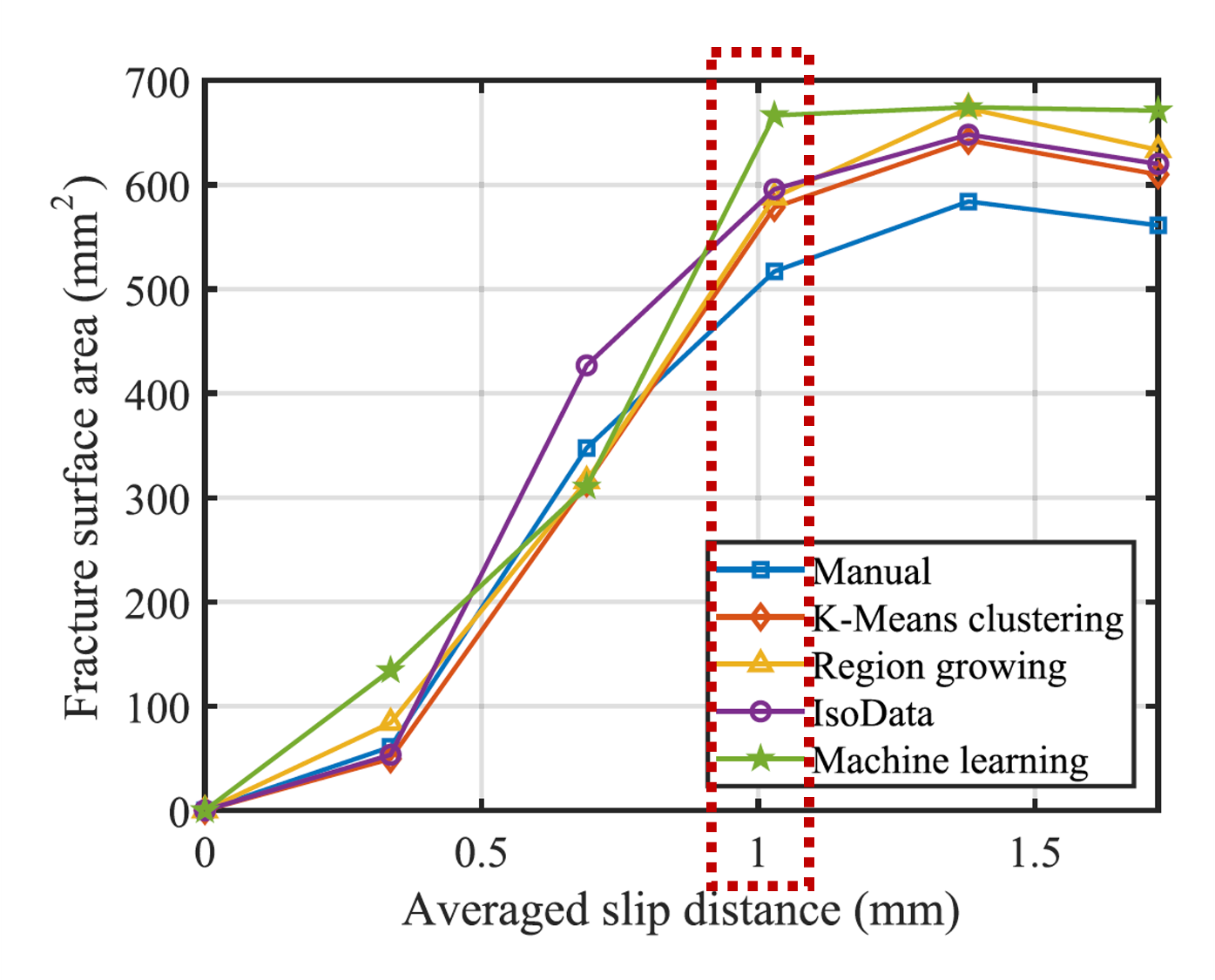
The Gaussian kernal used in Voorn et al. (2013) is isotropic. We anticipate that using an anisotropic Gaussian kernal has the potential of providing additional information about the fractures. The shape of an isosurface an anisotropic Gaussian kernal, i.e. 3D contour, yields a shape that has three principal directions of different magnitudes. This information can be utilized in identifying fracture orientation. This feature of fractures is helpful in understanding the extent of the fracture contribution in fluid flow within a rock sample.

Figure 8. Surface area calculations of the same images. After (Zhao et al., 2020, figure 4)

REFERENCES

Ketcham, R. A., et al., 2010, Three-dimensional measurement of fractures in heterogeneous materials using high-resolution X-ray computed tomography: Geosphere, 6, 499–514.

Lindeberg, T., 1998, Edge detection and ridge detection with automatic scale selection: 30, 117–154.

Voorn, M., et al., 2013, Multiscale Hessian fracture filtering for the enhancement and segmentation of narrow fractures in 3D image data: Computers and Geosciences, 57, 44–53.

Zhao, Q., S. D. Glaser, N. Tisato, and G. Grasselli, 2020, Assessing energy budget of laboratory fault slip using rotary shear experiments and micro-computed tomography: Geophysical Research Letters, 47, e2019GL084787.

Zhao, Q., N. Tisato, and G. Grasselli, 2017, Rotary shear experiments under X-ray micro-computed tomography: Review of Scientific Instruments, 88.

Zimmerman, R. W., and G. S. Bodvarsson, 1996, Hydraulic conductivity of rock fractures: Transport in Porous Media, 23, 1–30.

Zoback, M. D., 2007, Reservoir Geomechanics: Cambridge University Press.