



Software for the Practical Analysis of Materials

Olga Stamati: olga.stamati@3sr-grenoble.fr

ISRD-RCN Workshop:

*“Exploring Dynamic Properties of Earth and Planetary Materials
Using Neutron and X-Ray Methods”*



Grenoble, 23/05/2024

What is  ?

Let's play with some images

General information

What is **Spam** ?

Spam
ractical analysis

python library



(with some bits of C++)

python library



(with some bits of C++)

data analysis



(not AI enhanced!)

python library



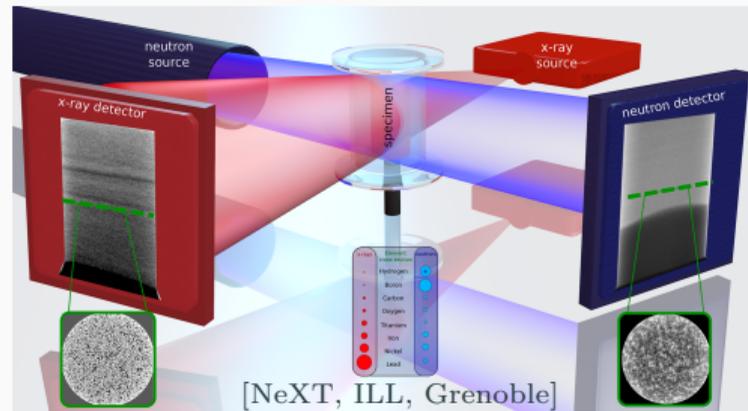
(with some bits of C++)

data analysis



(not AI enhanced!)

2D images and 3D volumes



X-ray & Neutron tomography
(for example)

python library



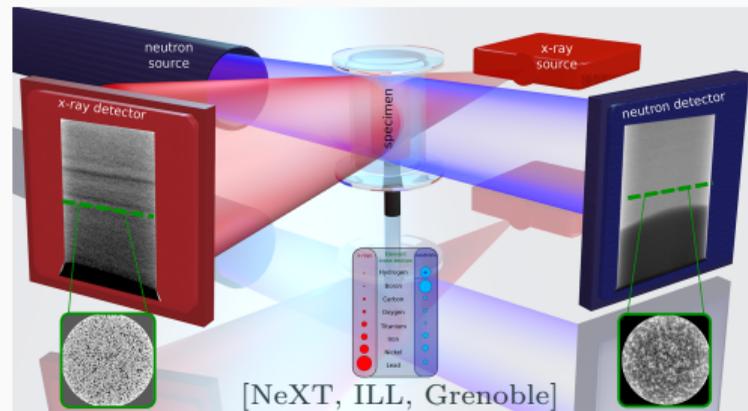
(with some bits of C++)

data analysis



(not AI enhanced!)

2D images and 3D volumes

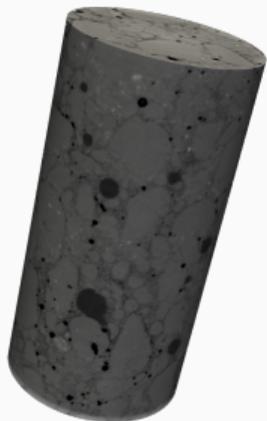


X-ray & Neutron tomography
(for example)

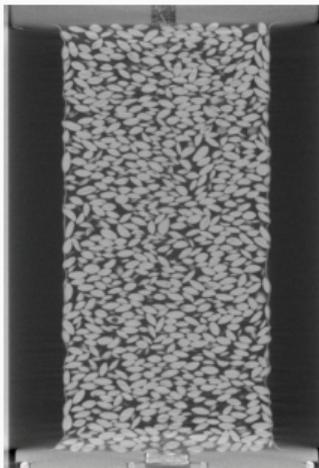
Strictly open-source !

Why do we need a software like **Spam** ractical analysis ?

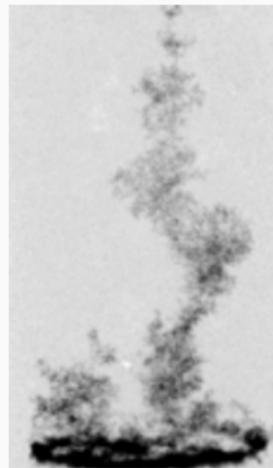
Concrete



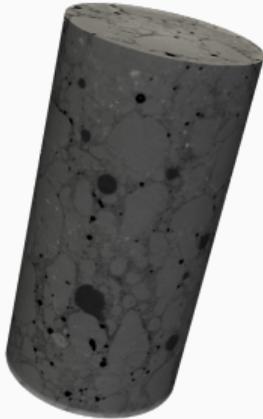
Lentils



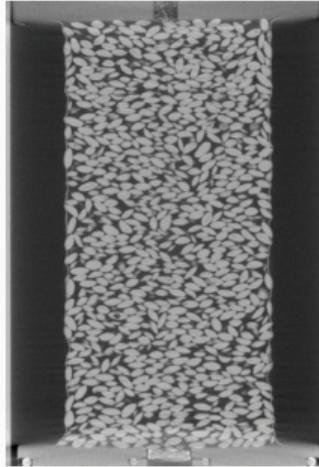
Sandstone



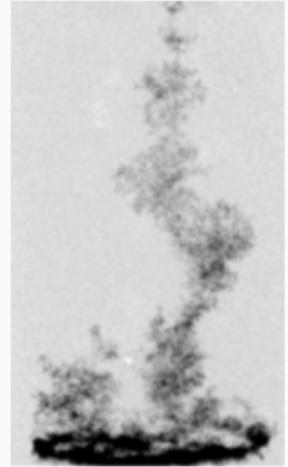
Concrete



Lentils



Sandstone

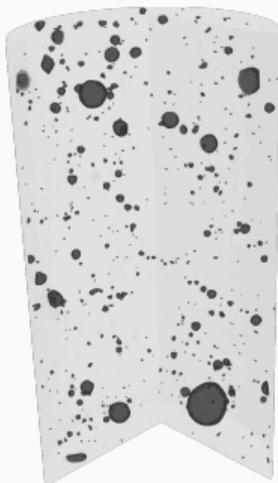


MORPHOLOGY

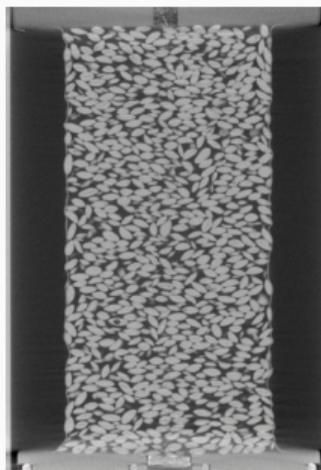
- Phase identification
- Volume fractions
- Size, shape
- Orientation, fabric
- Tortuosity, connectivity

Why do we need a software like **Spam** ractical analysis ?

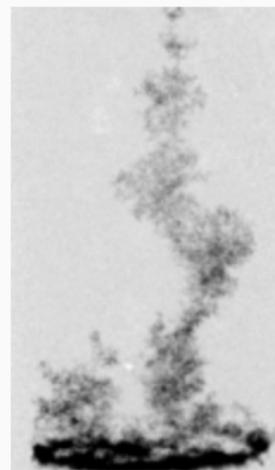
Concrete



Lentils



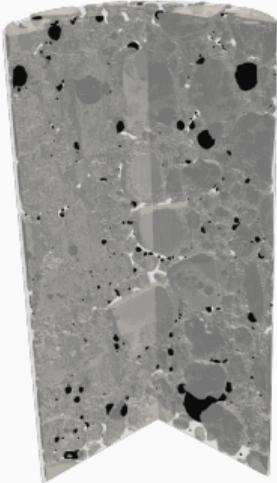
Sandstone



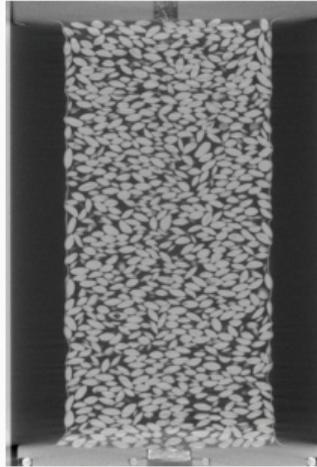
MORPHOLOGY

- Phase identification
- Volume fractions
- Size, shape
- Orientation, fabric
- Tortuosity, connectivity

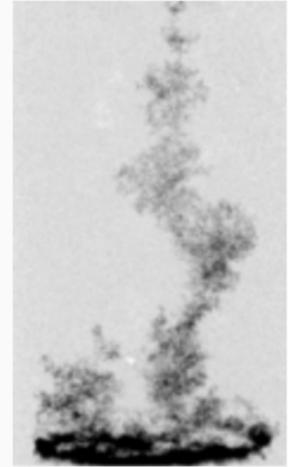
Concrete



Lentils



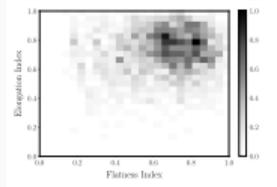
Sandstone



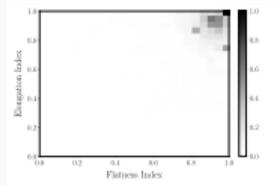
MORPHOLOGY

- Phase identification
- Volume fractions
- Size, shape
- Orientation, fabric
- Tortuosity, connectivity

Concrete

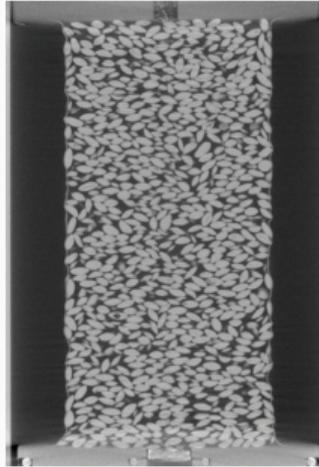


(a) aggregates

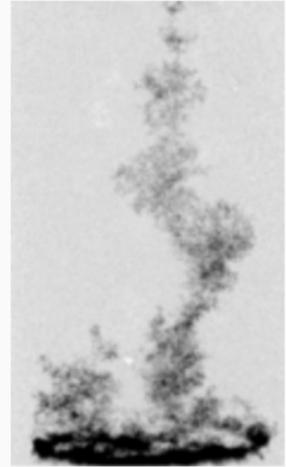


(b) macro-pores

Lentils



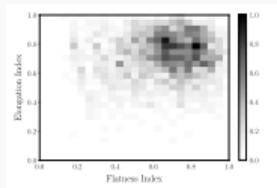
Sandstone



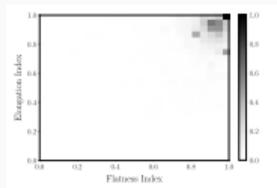
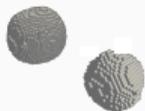
MORPHOLOGY

- Phase identification
- Volume fractions
- Size, shape
- Orientation, fabric
- Tortuosity, connectivity

Concrete

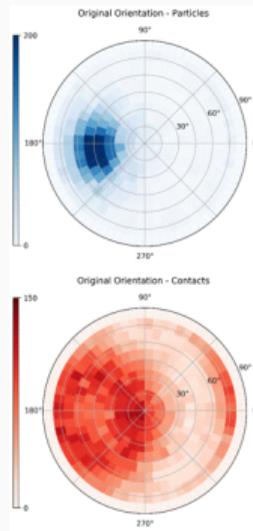


(a) aggregates



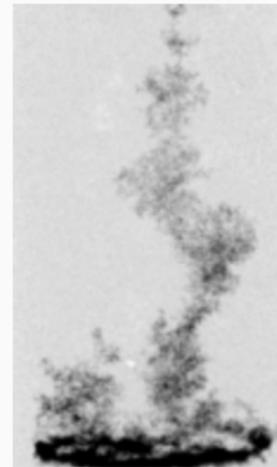
(b) macro-pores

Lentils



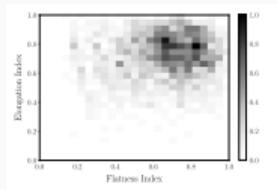
- Phase identification
- Volume fractions
- Size, shape
- Orientation, fabric
- Tortuosity, connectivity

Sandstone

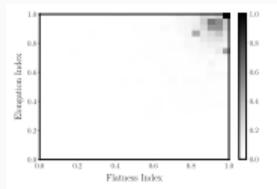


MORPHOLOGY

Concrete

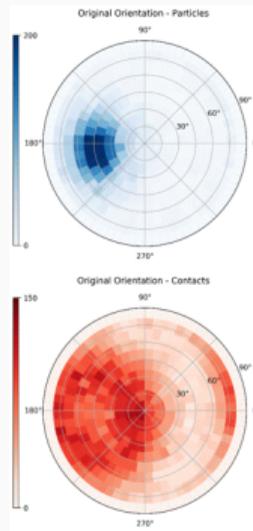


(a) aggregates

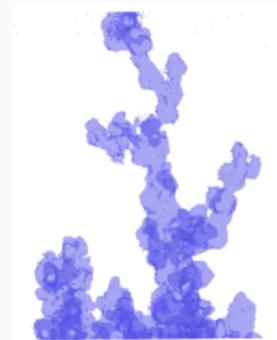


(b) macro-pores

Lentils



Sandstone



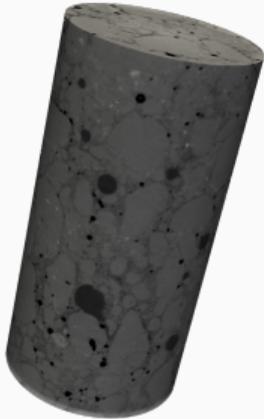
MORPHOLOGY

- Phase identification
- Volume fractions
- Size, shape
- Orientation, fabric
- Tortuosity, connectivity

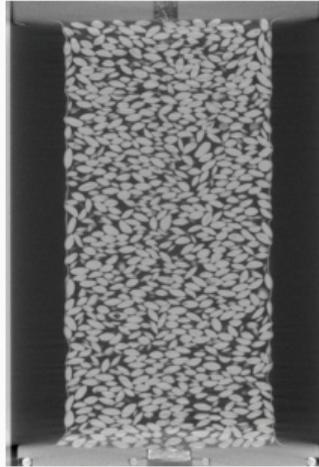
Why do we need a software like



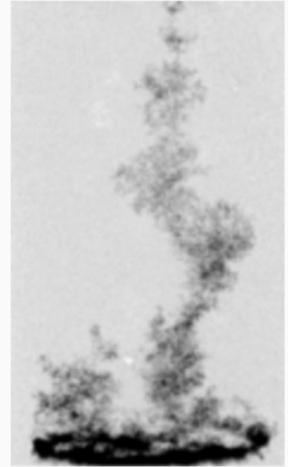
Concrete



Lentils



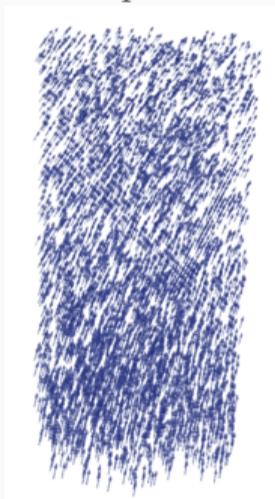
Sandstone



MORPHOLOGICAL
EVOLUTION

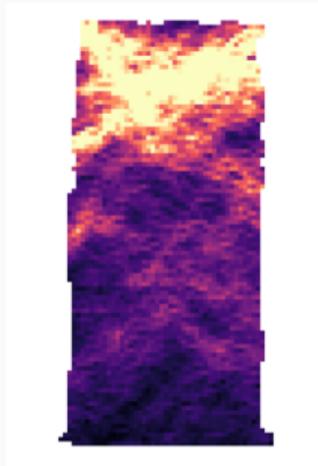
- δ (Volume fractions)
- δ (Size, shape)
- δ (Orientation, fabric)
- δ (Tortuosity, connectivity)
- **KINEMATIC FIELDS**

Concrete displacement field



[Stamati O. (2021) Cem. Concr. Research]

Lentils strain field



[Pinzón G. (2023) Gran. Matter]

Flow in sandstone

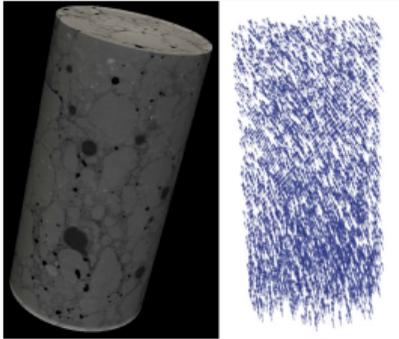


[Couture C. (in preparation)]

MORPHOLOGICAL
EVOLUTION

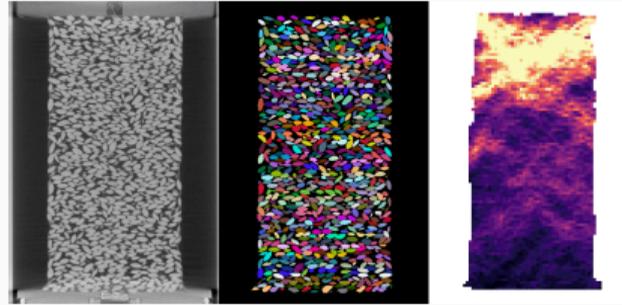
- δ (Volume fractions)
- δ (Size, shape)
- δ (Orientation, fabric)
- δ (Tortuosity, connectivity)
- **KINEMATIC FIELDS**

Concrete fracture



[Stamati O. (2021) Cem. Concr. Res.]

Particle tracking



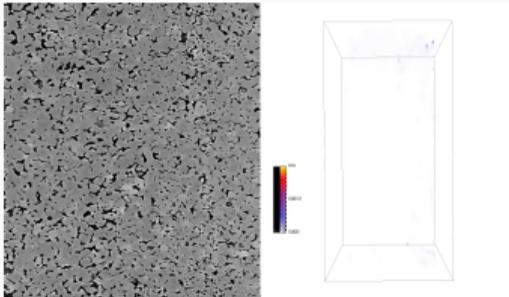
[Pinzón G. (2023) Gran. Matter]

Textile



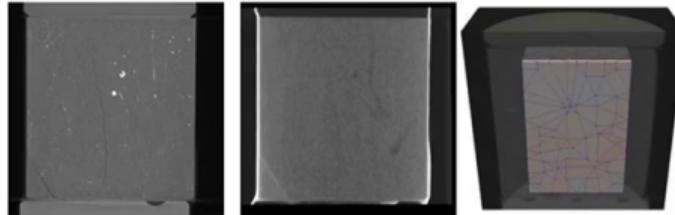
[Stamati O. (2023) Comp. Part A]

Sandstone failure



[Cartwright-Taylor A. (2022) Nature Comm.]

Claystone water absorption



[Stavropoulou E. (2020) Front. in Earth Science]

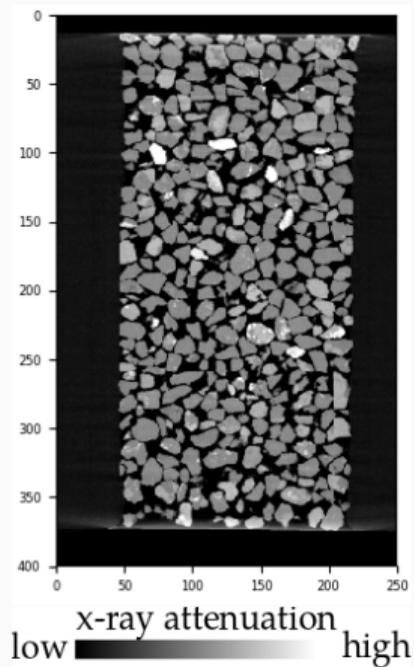
Tendon/Bone Interface



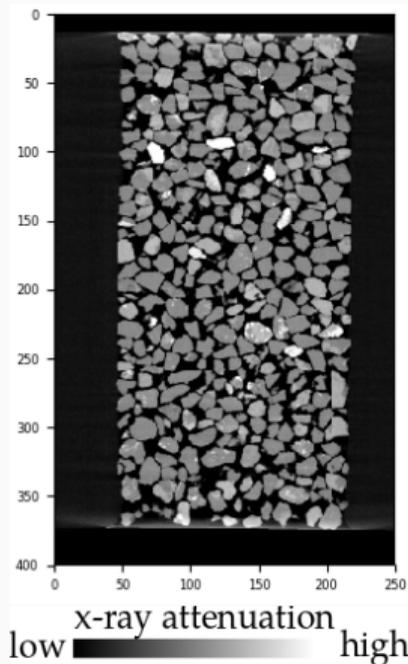
[Sensini A. (in preparation)]

Let's play with some images

Inside an X-ray image

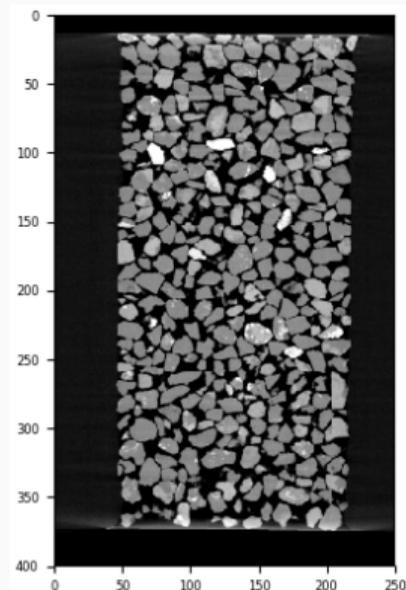


Inside an X-ray image



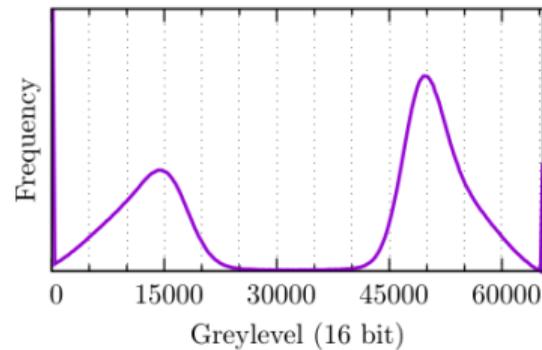
```
array([[33171, 29859, 31265, 28414, 30618, 31769, 0, 0, 0,  
16241, 29112, 29486],  
[28666, 27491, 31498, 28393, 31080, 22877, 0, 775, 43,  
34, 27448, 28065],  
[31150, 33212, 32147, 29513, 28288, 761, 261, 0, 0,  
2, 7497, 28411],  
[32261, 29453, 30281, 13581, 137, 0, 10, 0, 0, 0,  
45, 33, 31967],  
[28051, 12218, 152, 0, 34, 5, 216, 0, 0, 0,  
12, 265, 31143],  
[ 0, 0, 1462, 0, 118, 0, 1129, 636, 205,  
0, 65, 30024],  
[ 0, 0, 0, 120, 66, 0, 37, 10, 0, 0,  
1064, 0, 29666],  
[ 0, 1, 64, 0, 33, 11, 0, 1303, 0, 0,  
0, 62, 28679],  
[ 0, 0, 0, 7, 20, 0, 74, 47, 0, 0,  
5069, 2979, 28121],  
[ 1, 0, 0, 301, 13251, 25170, 17127, 26551, 27833,  
26803, 18042, 19713],  
[ 16, 0, 22134, 30711, 30216, 30021, 29950, 29447, 27158,  
29307, 24827, 9147],  
[ 0, 866, 30879, 32769, 28166, 29146, 31096, 31947, 30906,  
30306, 27292, 0]], dtype=uint16)
```

Inside an X-ray image

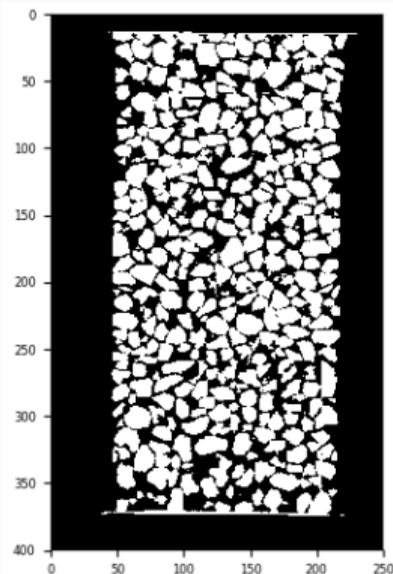


low  high
x-ray attenuation

```
array([[33171, 29859, 31265, 28414, 30618, 31769, 0, 0, 0,
16241, 29112, 29486],
[28666, 27491, 31498, 28393, 31080, 22877, 0, 775, 43,
34, 27448, 28065],
[31150, 33212, 32147, 29513, 28288, 761, 261, 0, 0,
2, 7497, 28411],
[32261, 29453, 30281, 13581, 137, 0, 10, 0, 0,
45, 33, 31967],
[28051, 12218, 152, 0, 34, 5, 216, 0, 0,
12, 265, 31143],
[ 0, 0, 1462, 0, 118, 0, 1129, 636, 205,
0, 65, 30024],
[ 0, 0, 0, 120, 66, 0, 37, 10, 0,
1064, 0, 29666],
[ 0, 1, 64, 0, 33, 11, 0, 1303, 0,
0, 62, 28679],
[ 0, 0, 0, 7, 20, 0, 74, 47, 0,
5069, 2979, 28121],
[ 1, 0, 0, 301, 13251, 25170, 17127, 26551, 27833,
26803, 18042, 19713],
[ 16, 0, 22134, 30711, 30216, 30021, 29950, 29447, 27158,
29307, 24827, 9147],
[ 0, 866, 30879, 32769, 28166, 29146, 31096, 31947, 30906,
30306, 27292, 0]], dtype=uint16)
```

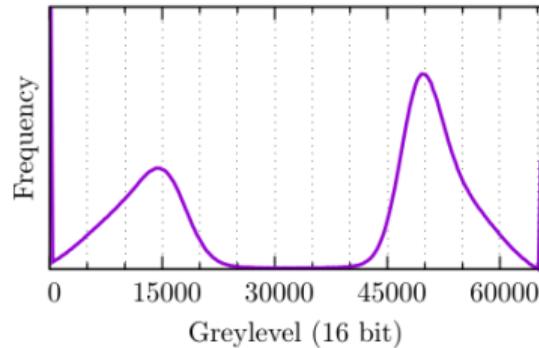


Inside an X-ray image

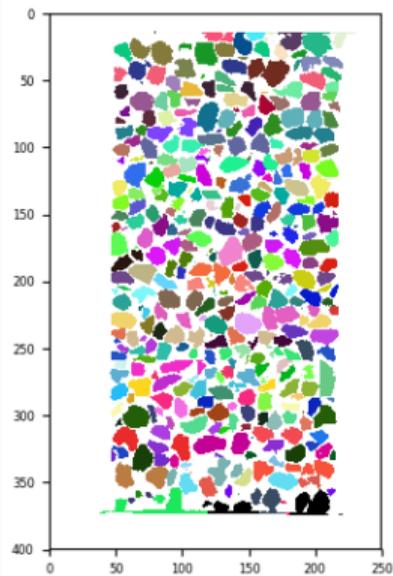


Binarised image

```
array([[1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1],  
       [1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1],  
       [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1],  
       [1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1],  
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],  
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],  
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],  
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],  
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],  
       [0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1],  
       [0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0],  
       [0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0]], dtype=uint8)
```



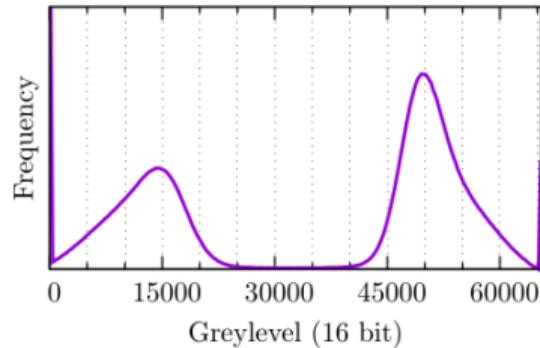
Inside an X-ray image



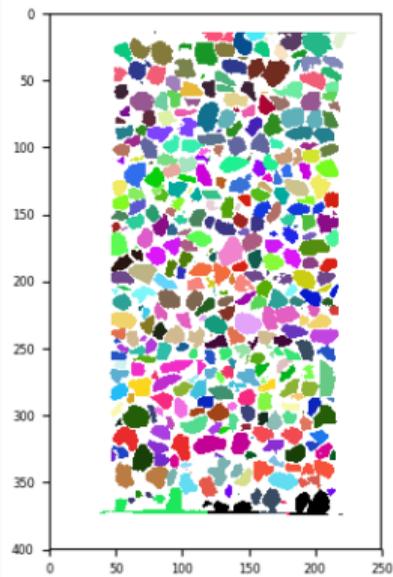
Labelled image



```
[[1583 1583 1583 1583 1583 1583 0 0 0 0 1614 1614]
 [1583 1583 1583 1583 1583 1583 0 0 0 0 1614 1614]
 [1583 1583 1583 1583 1583 0 0 0 0 0 0 1614]
 [1583 1583 1583 0 0 0 0 0 0 0 0 1614]
 [1583 0 0 0 0 0 0 0 0 0 0 1614]
 [ 0 0 0 0 0 0 0 0 0 0 0 1614]
 [ 0 0 0 0 0 0 0 0 0 0 0 1614]
 [ 0 0 0 0 0 0 0 0 0 0 0 1614]
 [ 0 0 0 0 0 0 0 0 0 0 0 1614]
 [ 0 0 0 0 0 1598 0 1598 1598 1598 1598 1614]
 [ 0 0 1598 1598 1598 1598 1598 1598 1598 1598 0]
 [ 0 0 1598 1598 1598 1598 1598 1598 1598 1598 0]]
```



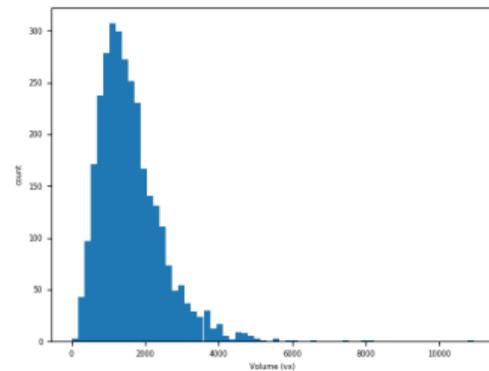
Inside an X-ray image



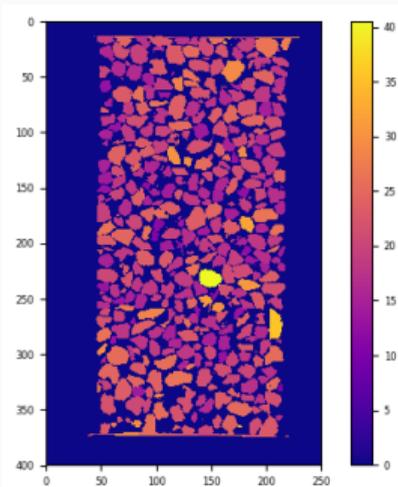
Labelled image



```
[1583 1583 1583 1583 1583 1583 0 0 0 0 1614 1614]
[1583 1583 1583 1583 1583 1583 0 0 0 0 1614 1614]
[1583 1583 1583 1583 1583 0 0 0 0 0 0 1614]
[1583 1583 1583 0 0 0 0 0 0 0 0 1614]
[1583 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 1598 0 1598 1598 1598 1598 1614]
[ 0 0 1598 1598 1598 1598 1598 1598 1598 1598 1598 0]
[ 0 0 1598 1598 1598 1598 1598 1598 1598 1598 1598 0]
```



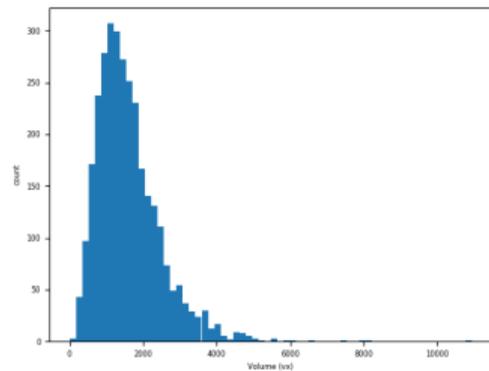
Inside an X-ray image



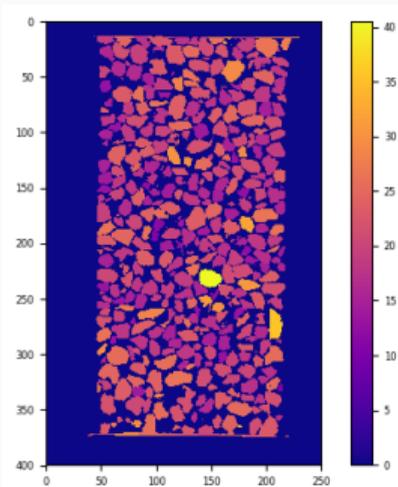
Particle volumes image



```
[1583 1583 1583 1583 1583 1583 0 0 0 0 1614 1614]
[1583 1583 1583 1583 1583 1583 0 0 0 0 1614 1614]
[1583 1583 1583 1583 1583 0 0 0 0 0 0 1614]
[1583 1583 1583 0 0 0 0 0 0 0 0 1614]
[1583 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 1598 0 1598 1598 1598 1598 1614]
[ 0 0 1598 1598 1598 1598 1598 1598 1598 1598 1598 0]
[ 0 0 1598 1598 1598 1598 1598 1598 1598 1598 1598 0]]
```



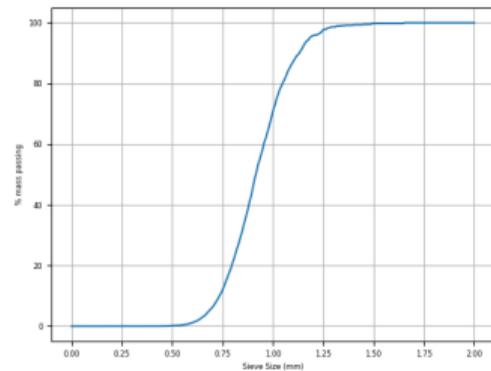
Inside an X-ray image



Particle volumes image

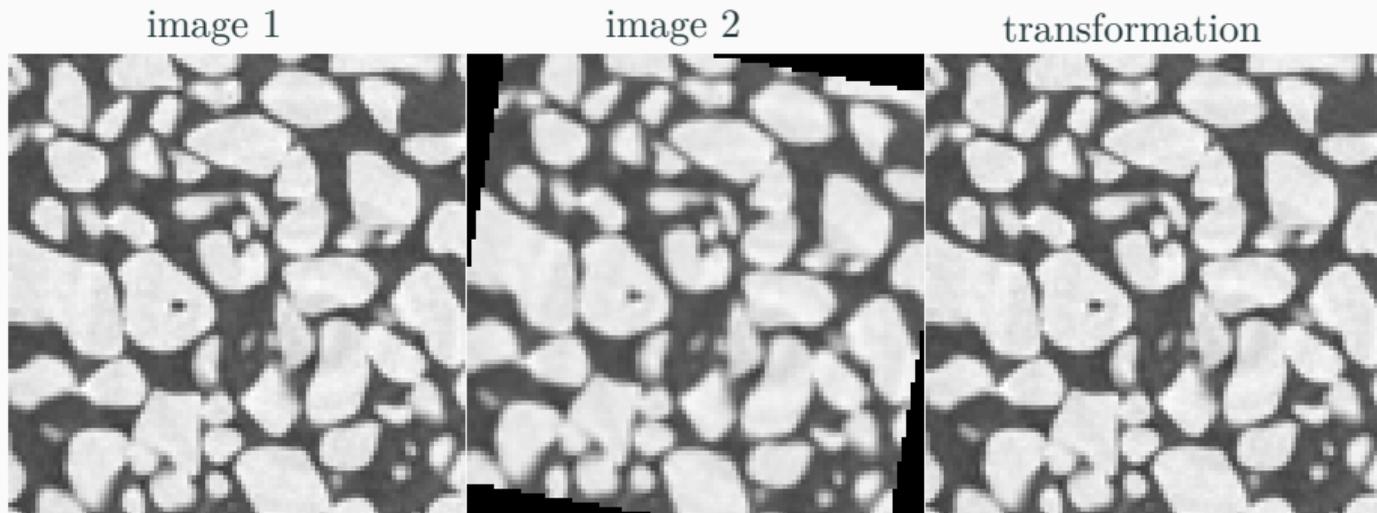


```
[1583 1583 1583 1583 1583 1583 0 0 0 0 1614 1614]
[1583 1583 1583 1583 1583 1583 0 0 0 0 1614 1614]
[1583 1583 1583 1583 1583 0 0 0 0 0 0 1614]
[1583 1583 1583 0 0 0 0 0 0 0 0 1614]
[1583 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 0 0 0 0 0 0 1614]
[ 0 0 0 0 0 1598 0 1598 1598 1598 1598 1614]
[ 0 0 1598 1598 1598 1598 1598 1598 1598 1598 0]
[ 0 0 1598 1598 1598 1598 1598 1598 1598 1598 0]]
```



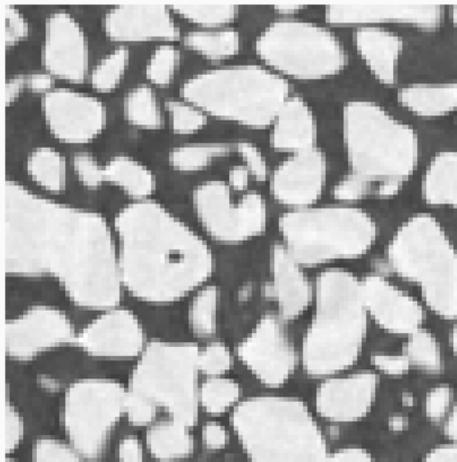
```
1 # import modules
2 import numpy, tifffile
3 import spam.label, spam.plotting
4
5 # 1. load image and plot the grey level histogram
6 im = tifffile.imread("grains.tif")
7 spam.plotting.plotGreyLevelHistogram(grey, showGraph=True)
8 # 2. identification of grains
9 binary = im >= 18000
10 # 3. labelling of grains
11 labelled = spam.label.watershed(binary)
12 # 4. volumes calculation
13 volumes = spam.label.volumes(labelled)
14 plt.hist(volumes, bins=64); plt.xlabel("Volume (vx)"); plt.ylabel("count"); plt.show()
15 # 5. plot particle size distribution
16 radii = spam.label.equivalentRadii(labelled)
17 spam.plotting.plotParticleSizeDistribution(radii*15*4/1000.0, bins=256, units="mm")
18 # 6. visualise the grains coloured with volumes
19 labelledMaxVol = spam.label.convertLabelToFloat(labelled, volumes)
20 plt.imshow(labelledMaxVol[:, :, labelledMaxVol.shape[2]//2], cmap="plasma"); plt.show()
```

What about the evolution through time?



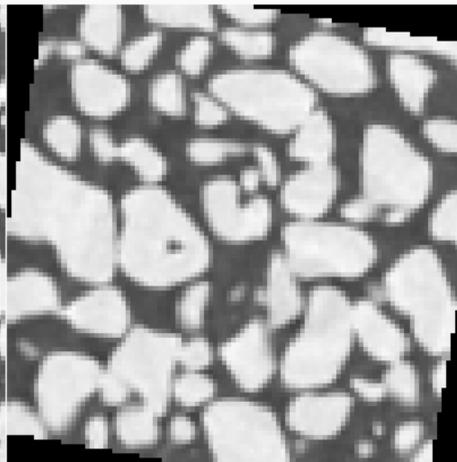
What about the evolution through time?

image 1



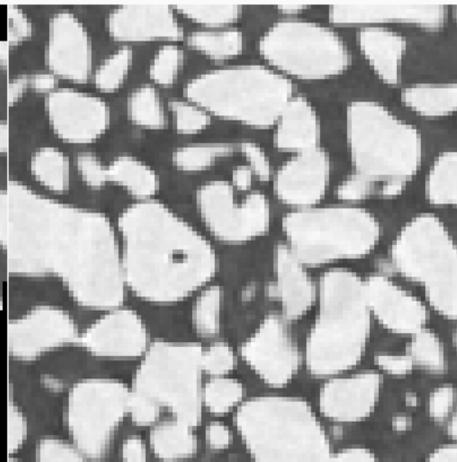
```
array([[11831, 11145, 11580, 12270, 12640, 12514, 13176, 14835, 20326,
        29225, 34191, 21655, 12283],
       [18759, 12223, 12034, 12864, 13021, 12939, 12926, 16789, 23822,
        29928, 33987, 26894, 13810],
       [27971, 15814, 12261, 12486, 13352, 12963, 12321, 17037, 25474,
        31316, 33720, 31330, 17174],
       [30087, 18643, 12490, 13615, 13259, 12608, 12316, 17200, 26466,
        31431, 32565, 32627, 24529],
       [30940, 20799, 13864, 13810, 12368, 12428, 12638, 14913, 22127,
        28696, 30402, 31378, 29082],
       [30308, 20619, 13442, 13171, 12174, 11780, 13367, 15067, 10360,
        20400, 23084, 24087, 21327],
       [28231, 10996, 12774, 11774, 10907, 11451, 17046, 24109, 26307,
        18931, 12581, 12947, 11612],
       [20042, 11917, 12057, 11458, 10142, 12744, 21917, 30117, 32083,
        26414, 13083, 9227, 9560],
       [11030, 10643, 11377, 11306, 9810, 12978, 24110, 30924, 32738,
        28492, 14357, 9786, 9021],
       [10034, 10527, 11255, 11807, 10928, 11390, 18715, 25353, 27755,
        21958, 12138, 10322, 10209],
       [ 9231, 9982, 11295, 11915, 11880, 11152, 12225, 15596, 16983,
        12342, 10224, 10482, 11079],
       [10336, 9713, 11265, 12404, 11966, 11322, 11061, 10848, 10400,
        9032, 9688, 11002, 11866],
       [13713, 11509, 12580, 13535, 11586, 10646, 10546, 10882, 10976,
        10456, 9832, 9996, 10933]], dtype=uint16)
```

image 2



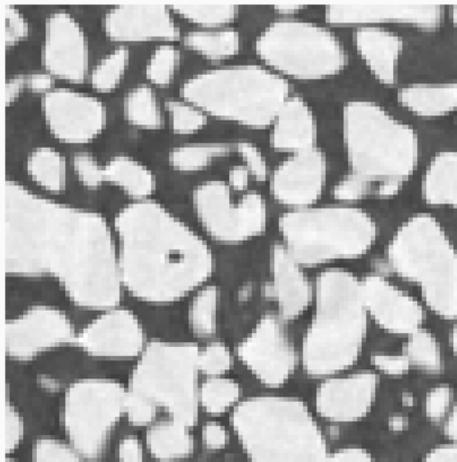
```
array([[32738, 20492, 14357, 9786, 9831, 10480, 15689, 17897, 15056,
        11978, 11473, 11109, 11030],
       [27755, 21958, 12138, 10322, 10209, 9788, 12119, 16130, 15933,
        13200, 12479, 11724, 11014],
       [16983, 12342, 10224, 10482, 11079, 10295, 11008, 14399, 16311,
        14115, 12619, 11464, 10091],
       [10400, 9032, 9008, 11002, 11066, 10993, 11439, 12952, 15983,
        10938, 19381, 21610, 10468],
       [10976, 10456, 9832, 9996, 10933, 11927, 11476, 11817, 13966,
        20464, 28899, 30405, 28835],
       [11100, 10816, 10411, 10206, 10400, 10965, 11633, 12334, 12732,
        15483, 29714, 31859, 30801],
       [17810, 11712, 10137, 10024, 11086, 10827, 11452, 11589, 11653,
        16397, 26023, 29611, 26193],
       [32845, 26023, 17541, 11582, 11310, 12237, 12162, 12005, 11463,
        12732, 18506, 20914, 10539],
       [33943, 33966, 32107, 21946, 11953, 13404, 13231, 13101, 12433,
        12164, 13428, 13824, 12103],
       [33859, 32898, 34371, 32940, 16905, 12897, 13213, 12454, 12711,
        12339, 12456, 13448, 13074],
       [33859, 32843, 32587, 34144, 24247, 12734, 12364, 11712, 11527,
        12905, 13032, 13041, 12610],
       [37681, 28889, 31693, 34115, 25019, 12476, 12798, 10718, 11111,
        12500, 12814, 12958, 11950],
       [26569, 32452, 33644, 31245, 17666, 11383, 12677, 11976, 11876,
        12189, 12612, 12771, 11369]], dtype=uint16)
```

transformation



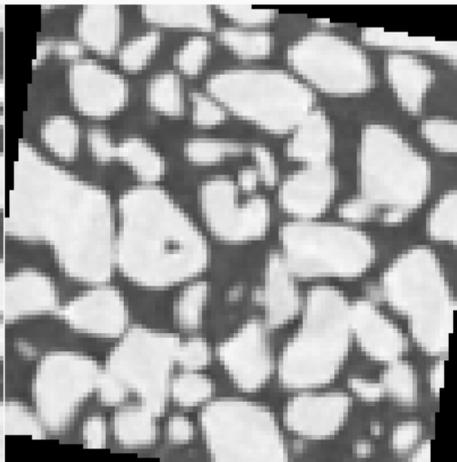
What about the evolution through time?

image 1



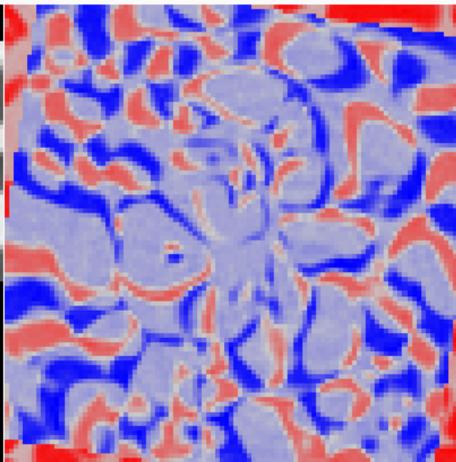
```
array([[11831, 11145, 11580, 12270, 12640, 12514, 13176, 14835, 20326,
        29225, 34191, 21655, 12283],
       [18759, 12223, 12034, 12864, 13021, 12939, 12926, 16789, 23822,
        29928, 33987, 26894, 13810],
       [27971, 15814, 12261, 12486, 13352, 12963, 12321, 17037, 25474,
        31316, 33720, 31330, 17174],
       [30087, 16643, 12490, 13615, 13259, 12608, 12316, 17200, 26466,
        31431, 32565, 32627, 24529],
       [30940, 20799, 13864, 13810, 12368, 12428, 12638, 14913, 22127,
        28696, 30402, 31378, 29082],
       [30308, 20619, 13442, 13171, 12174, 11780, 13367, 15067, 10360,
        20400, 23084, 24087, 21327],
       [28231, 10996, 12774, 11774, 10907, 11451, 17046, 24109, 26307,
        18911, 12581, 12947, 11612],
       [20042, 11917, 12057, 11458, 10142, 12744, 21917, 30117, 32083,
        26414, 13083, 9227, 9560],
       [11830, 10643, 11377, 11306, 9810, 12978, 24110, 30924, 32738,
        28492, 14357, 9786, 9021],
       [10034, 10527, 11255, 11807, 10928, 11390, 18715, 25353, 27755,
        21958, 12138, 10322, 10209],
       [ 9231, 9982, 11295, 11915, 11880, 11152, 12225, 15596, 16983,
        12342, 10224, 10482, 11079],
       [10336, 9713, 11265, 12484, 11966, 11322, 11061, 10848, 10408,
        9032, 9688, 11002, 11060],
       [13713, 11569, 12580, 13535, 11586, 10646, 10546, 10882, 10976,
        10456, 9832, 9996, 10933]], dtype=uint16)
```

image 2



```
array([[32738, 20492, 14357, 9786, 9831, 10480, 15689, 17897, 15056,
        11978, 11473, 11189, 11030],
       [27755, 21958, 12138, 10322, 10209, 9788, 12119, 16130, 15933,
        13200, 12479, 11724, 11014],
       [16983, 12342, 10224, 10482, 11079, 10295, 11008, 14399, 16311,
        14115, 12619, 11464, 10091],
       [10408, 9032, 9008, 11002, 11066, 10993, 11439, 12952, 15983,
        10938, 19381, 21610, 10468],
       [10976, 10456, 9832, 9996, 10933, 11927, 11476, 11817, 13966,
        20464, 28899, 30485, 28835],
       [11100, 10816, 10411, 10206, 10408, 10965, 11633, 12334, 12732,
        15483, 29714, 31859, 30801],
       [17810, 11712, 10137, 10024, 11086, 10827, 11452, 11589, 11053,
        16397, 26023, 29611, 26193],
       [32845, 26023, 17541, 11582, 11310, 12237, 12162, 12005, 11463,
        12752, 18506, 20914, 10539],
       [33943, 33966, 32107, 21946, 11953, 13404, 13231, 13101, 12433,
        12164, 13428, 13824, 12103],
       [33859, 32898, 34371, 32940, 10905, 12897, 13213, 12454, 12711,
        12339, 12456, 13448, 13074],
       [33859, 32843, 32587, 34144, 24247, 12734, 12364, 11712, 11527,
        12905, 13032, 13041, 12610],
       [37681, 32889, 31693, 34115, 25019, 12476, 12798, 10718, 11111,
        12500, 12814, 12958, 11950],
       [26569, 32452, 33644, 31245, 17666, 11383, 12677, 11976, 11876,
        12189, 12612, 12771, 11369]], dtype=uint16)
```

residual



```
array([[20007, 17347, 2849, 63052, 62727, 63502, 2513, 3062, 60266,
        40209, 42010, 55050, 64289],
       [ 8996, 9725, 104, 62594, 62724, 62305, 64729, 64077, 57647,
        40008, 44028, 58366, 62740],
       [54548, 62064, 63499, 63532, 63263, 62868, 64223, 62898, 56373,
        40355, 44435, 45670, 59653],
       [44957, 55925, 62734, 62923, 64143, 63921, 64659, 61280, 55053,
        51041, 52352, 54525, 60475],
       [45572, 55193, 61504, 61722, 64101, 65035, 64374, 62440, 57375,
        57304, 64033, 64643, 65288],
       [46328, 55533, 62505, 62571, 63762, 64713, 63802, 62003, 59908,
        64019, 6030, 7772, 9474],
       [51115, 40252, 62899, 64306, 179, 64912, 59942, 53016, 50882,
        63002, 13442, 16664, 14581],
       [12003, 14106, 5484, 124, 1108, 65029, 55781, 47424, 44910,
        51874, 5423, 11687, 6979],
       [22113, 23323, 20820, 10620, 2143, 426, 54057, 47713, 45231,
        49208, 64607, 4038, 2332],
       [23825, 22371, 23116, 21333, 5977, 1507, 00034, 52637, 50492,
        55917, 310, 3126, 2865],
       [24628, 22861, 21292, 22229, 12367, 1582, 139, 61652, 60080,
        563, 2008, 2559, 1531],
       [22345, 23176, 21428, 21631, 13053, 1154, 1737, 65406, 703,
        3528, 3126, 1956, 90],
       [12856, 20883, 21058, 17710, 6080, 737, 2131, 1094, 900,
        1733, 2780, 2775, 436]], dtype=uint16)
```


The general idea of Digital Volume Correlation (DVC)

DVC aims to find a **transformation** that links two 3D images $f(\mathbf{x})$ and $g(\mathbf{x})$.

The general idea of Digital Volume Correlation (DVC)

DVC aims to find a **transformation** that links two 3D images $f(\mathbf{x})$ and $g(\mathbf{x})$.

$$f(\mathbf{x}) - g(\Phi \cdot \mathbf{x}) = 0$$

A deformation function Φ is sought such that the material point in position \mathbf{x} in the reference image corresponds to the same material point in position $\mathbf{x}' = \Phi \cdot \mathbf{x}$ in the deformed image.

The general idea of Digital Volume Correlation (DVC)

DVC aims to find a **transformation** that links two 3D images $f(\mathbf{x})$ and $g(\mathbf{x})$.

$$f(\mathbf{x}) - g(\Phi \cdot \mathbf{x}) = 0$$

A deformation function Φ is sought such that the material point in position \mathbf{x} in the reference image corresponds to the same material point in position $\mathbf{x}' = \Phi \cdot \mathbf{x}$ in the deformed image.

Deformation function in 

$$\Phi = \begin{bmatrix} F_{zz} & F_{zy} & F_{zx} & t_z \\ F_{yz} & F_{yy} & F_{yx} & t_y \\ F_{xz} & F_{xy} & F_{xx} & t_x \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{with } \mathbf{x} = \begin{pmatrix} z \\ y \\ x \\ 1 \end{pmatrix} \quad \text{and } \mathbf{x}' = \Phi \cdot \mathbf{x}$$

where Φ is **linear** and **homogeneous** accounting for:
translation, rotation, normal and shear strain

Gradually **minimise** the SSD between the **reference** image f and the **deformed** image \tilde{g} corrected by a **trial** deformation function $\tilde{\Phi}$:

$$\mathcal{T}(\tilde{\Phi}) = \frac{1}{2} \sum_{\mathbf{x} \in \Omega} (f(\mathbf{x}) - g(\tilde{\Phi} \cdot \mathbf{x}))^2 \quad \text{with} \quad \tilde{\Phi} = \underset{\tilde{\Phi}}{\operatorname{argmin}} \mathcal{T}(\tilde{\Phi})$$

Gradually **minimise** the SSD between the **reference** image f and the **deformed** image \tilde{g} corrected by a **trial** deformation function $\tilde{\Phi}$:

$$\mathcal{T}(\tilde{\Phi}) = \frac{1}{2} \sum_{x \in \Omega} (f(x) - g(\tilde{\Phi} \cdot x))^2 \quad \text{with} \quad \tilde{\Phi} = \underset{\tilde{\Phi}}{\operatorname{argmin}} \mathcal{T}(\tilde{\Phi})$$

A Gauss-Newton iterative scheme is implemented based on a 1st order Taylor expansion of the updated deformed image \tilde{g} for **small** corrections of $\delta\tilde{\Phi}$:

$$g(\tilde{\Phi}^{(n+1)} \cdot x) = g(\tilde{\Phi}^{(n)} \cdot x) + \nabla g(\tilde{\Phi}^{(n)} \cdot x) \cdot \delta\tilde{\Phi}^{(n+1)} \cdot \tilde{\Phi}^{(n)} \cdot x$$

Gradually **minimise** the SSD between the **reference** image f and the **deformed** image \tilde{g} corrected by a **trial** deformation function $\tilde{\Phi}$:

$$\mathcal{T}(\tilde{\Phi}) = \frac{1}{2} \sum_{\mathbf{x} \in \Omega} (f(\mathbf{x}) - g(\tilde{\Phi} \cdot \mathbf{x}))^2 \quad \text{with} \quad \tilde{\Phi} = \underset{\tilde{\Phi}}{\operatorname{argmin}} \mathcal{T}(\tilde{\Phi})$$

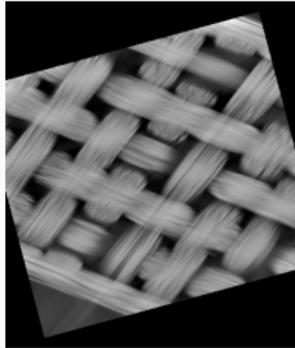
A Gauss-Newton iterative scheme is implemented based on a 1st order Taylor expansion of the updated deformed image \tilde{g} for **small** corrections of $\delta\tilde{\Phi}$:

$$g(\tilde{\Phi}^{(n+1)} \cdot \mathbf{x}) = g(\tilde{\Phi}^{(n)} \cdot \mathbf{x}) + \nabla g(\tilde{\Phi}^{(n)} \cdot \mathbf{x}) \cdot \delta\tilde{\Phi}^{(n+1)} \cdot \tilde{\Phi}^{(n)} \cdot \mathbf{x}$$

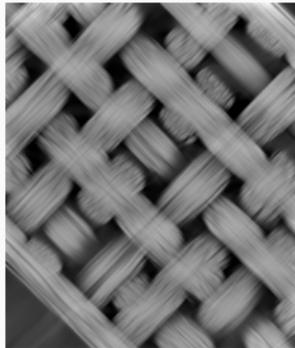
The system in a matrix-vector format:

$$\mathbf{M}^{(n)} \delta\tilde{\Phi}^{(n+1)} = \mathbf{A}^{(n)}$$

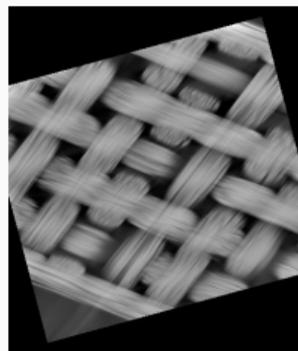
- \mathbf{M} is the Hessian: gradient of g and
- \mathbf{A} is the Jacobian: contains the difference between f and \tilde{g}



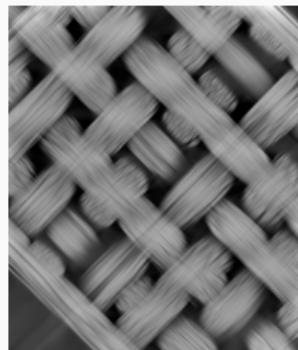
$g(x)$



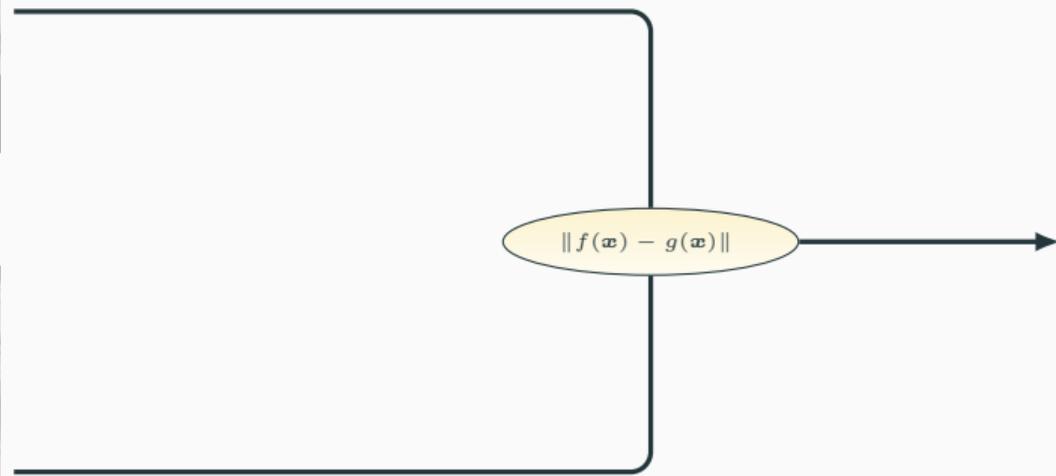
$f(x)$



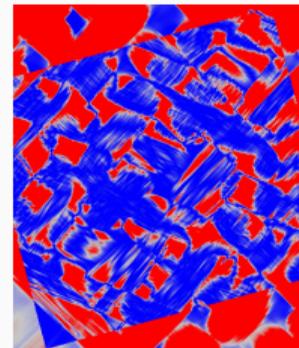
$g(x)$



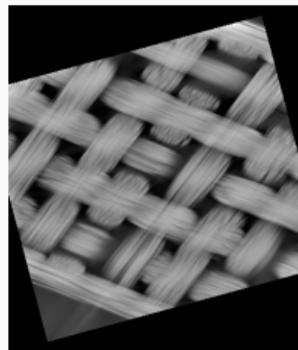
$f(x)$



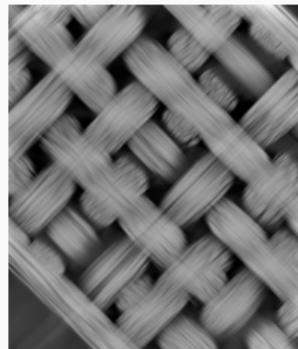
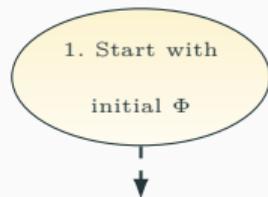
$\|f(x) - g(x)\|$



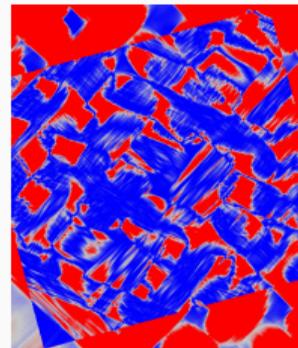
initial difference



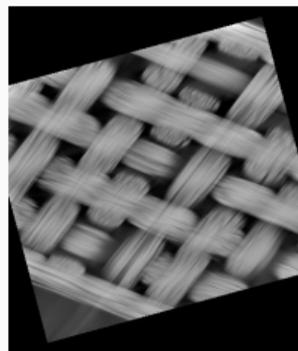
$g(x)$



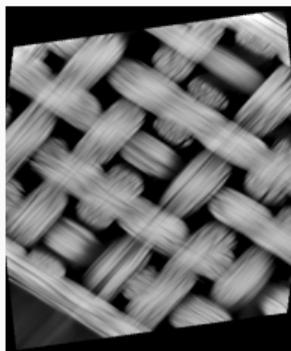
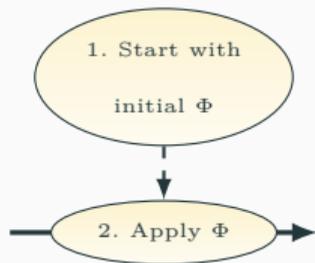
$f(x)$



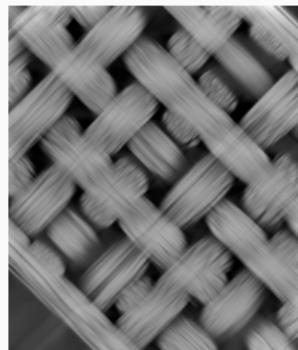
initial difference



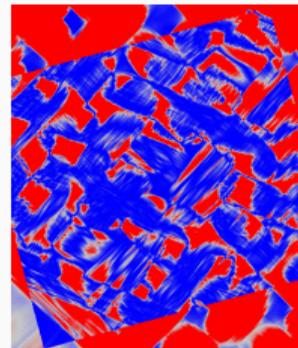
$g(x)$



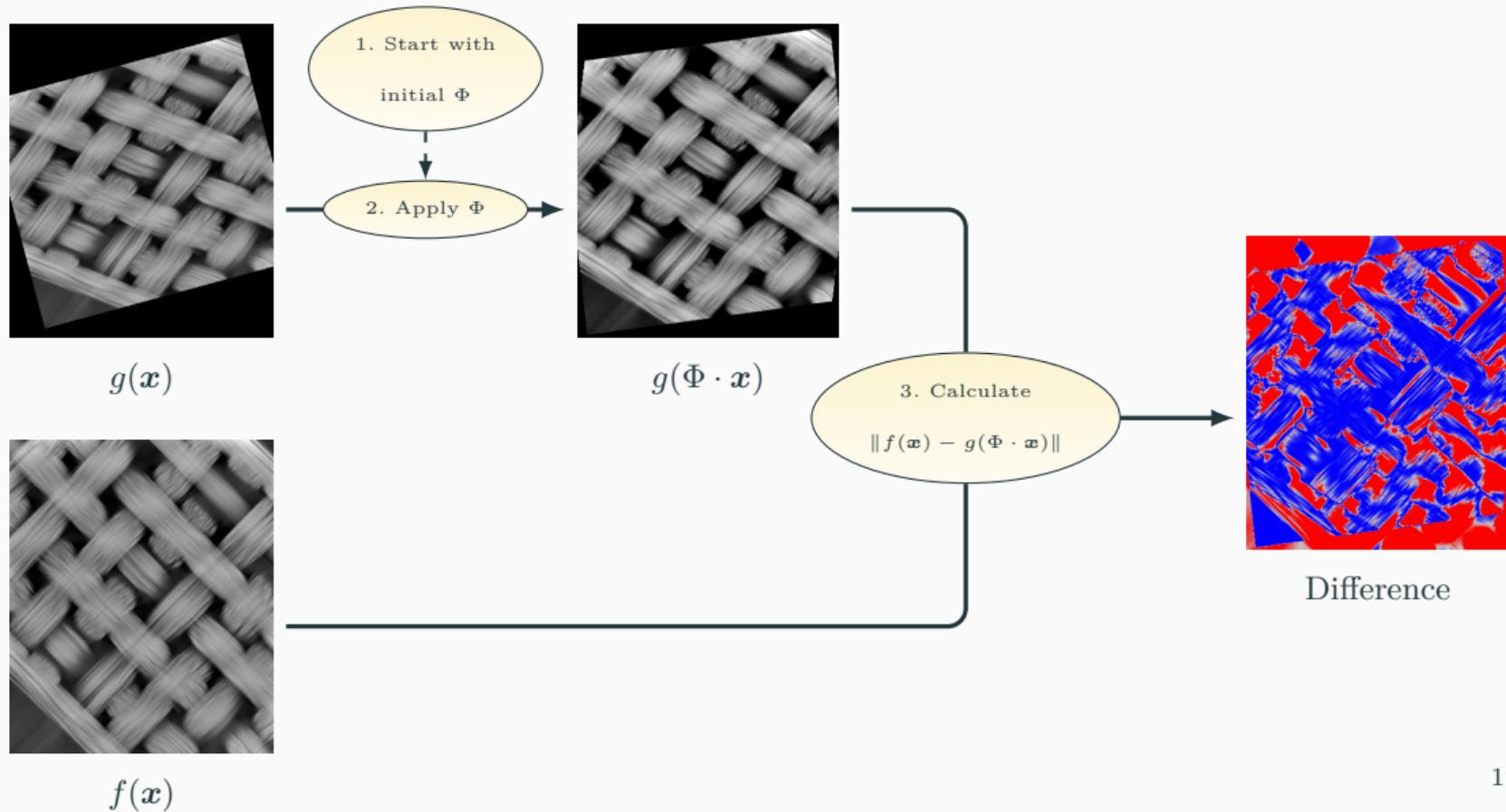
$g(\Phi \cdot x)$

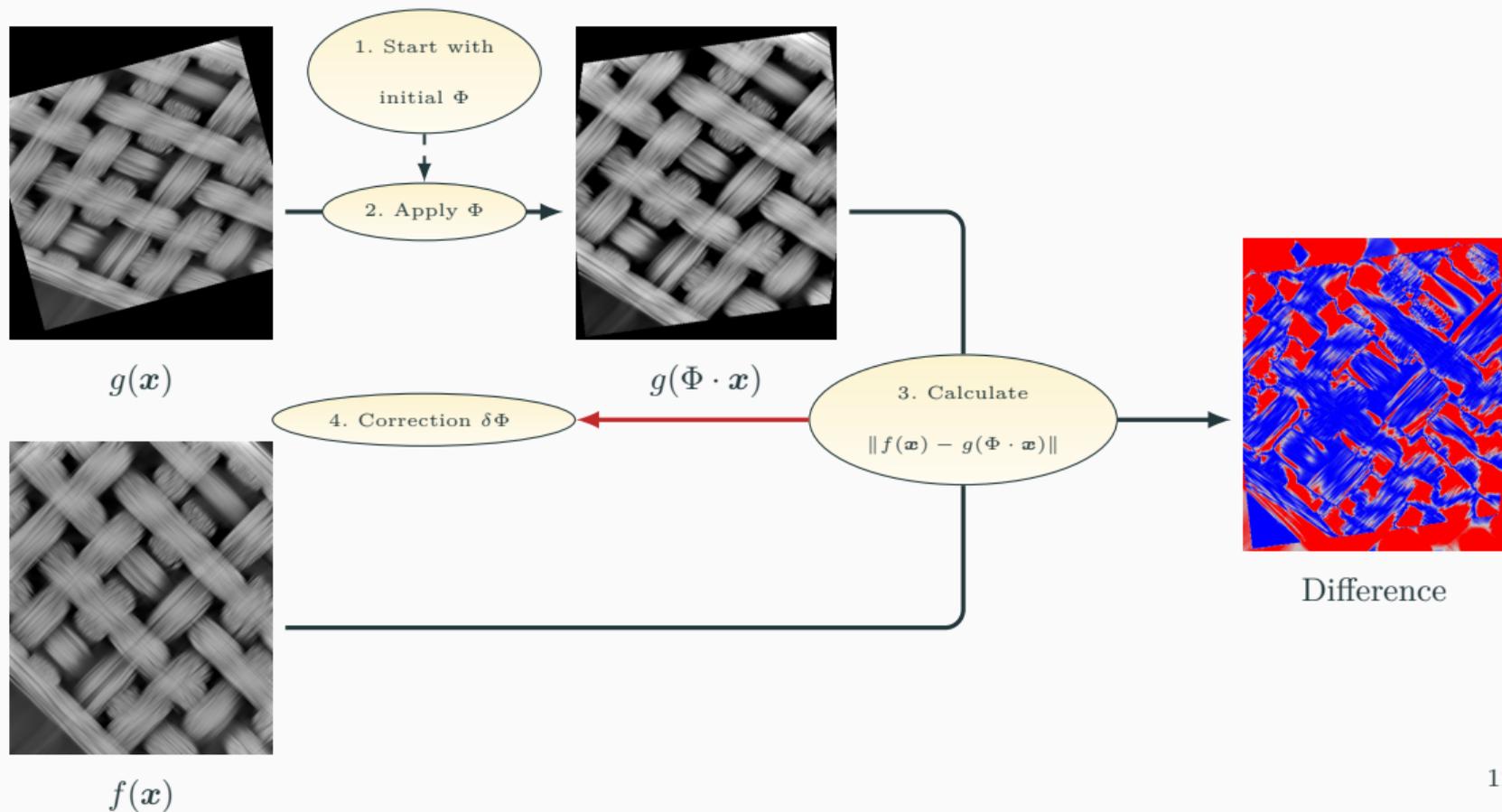


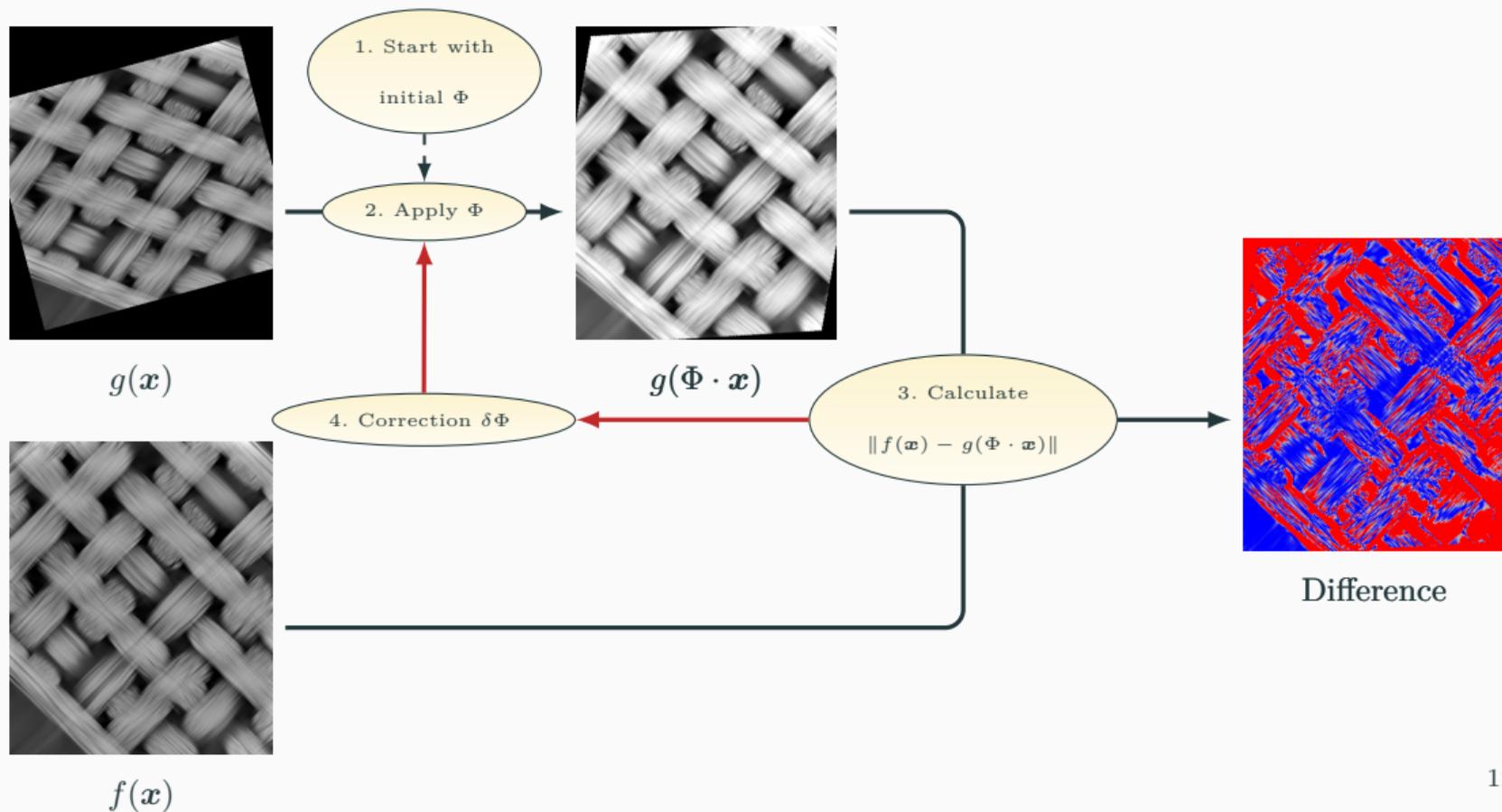
$f(x)$

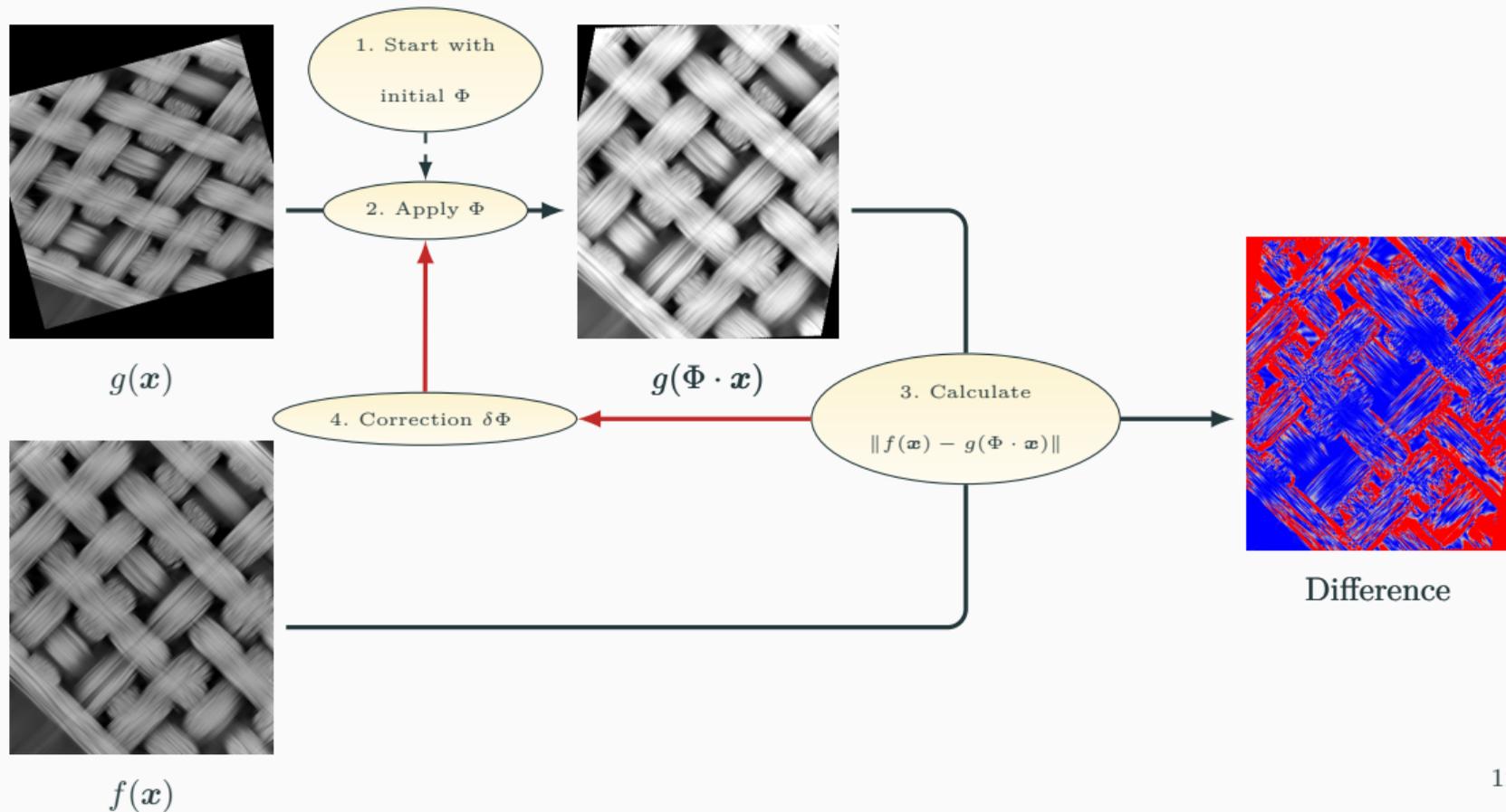


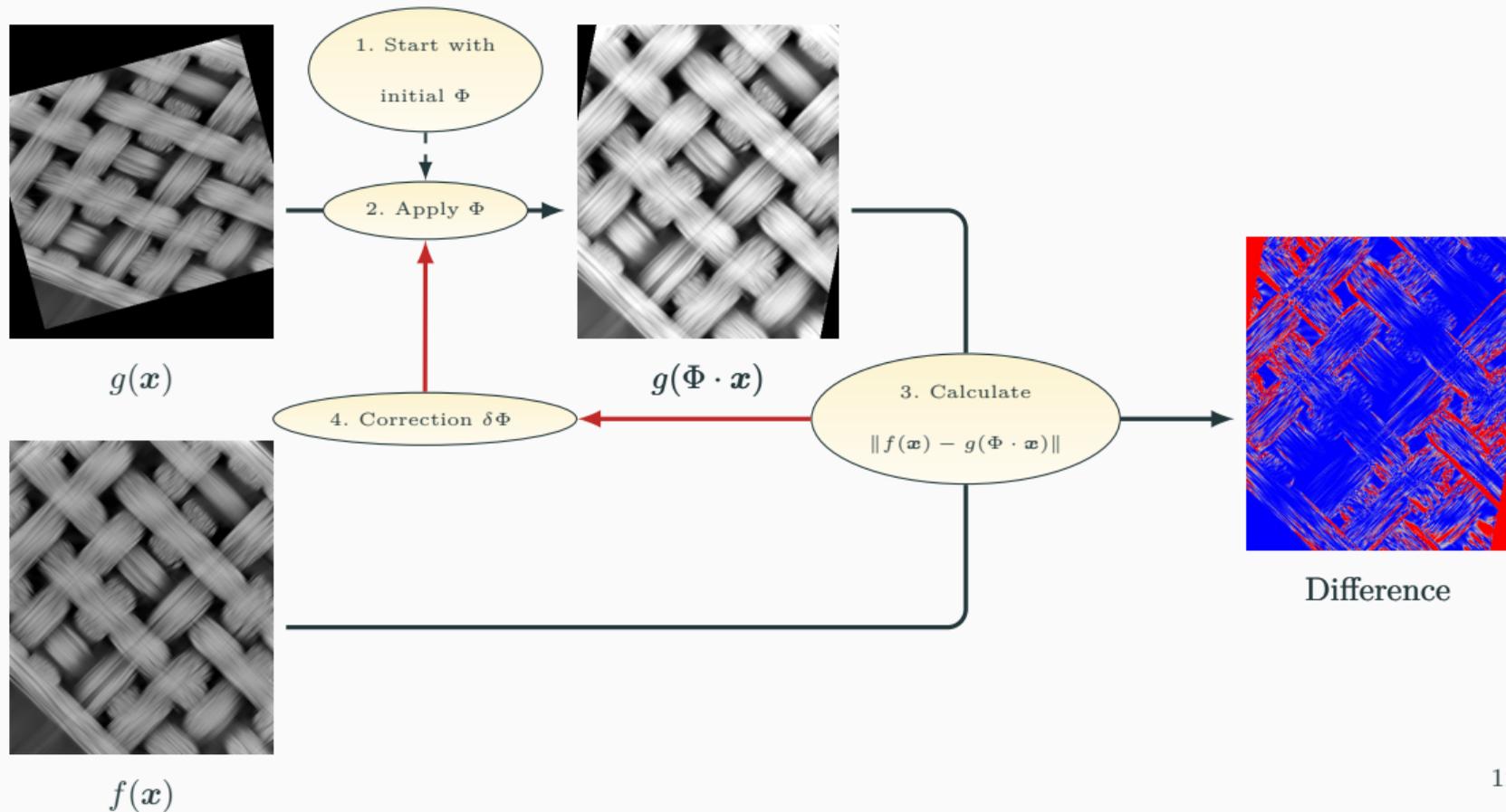
initial difference

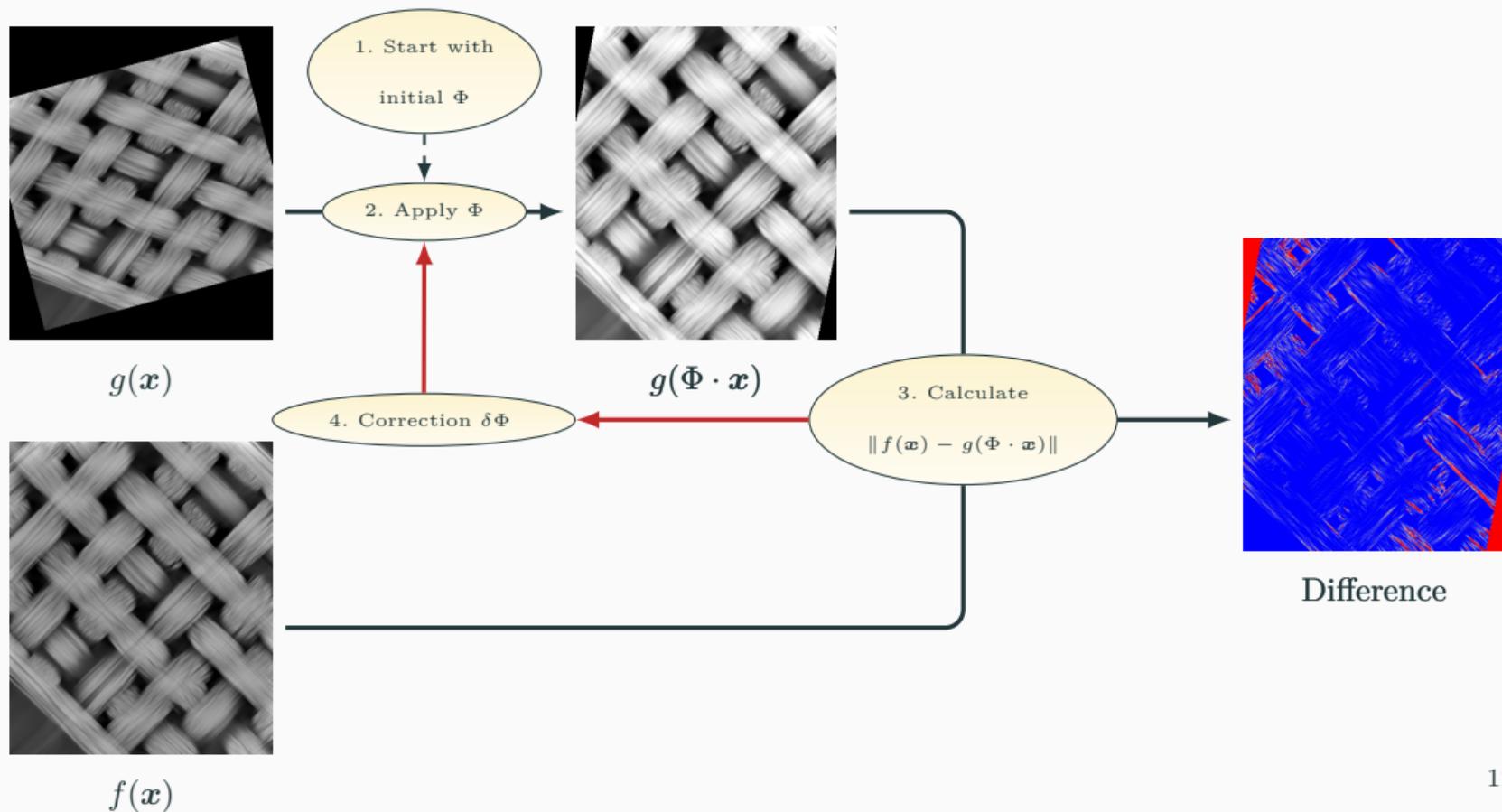


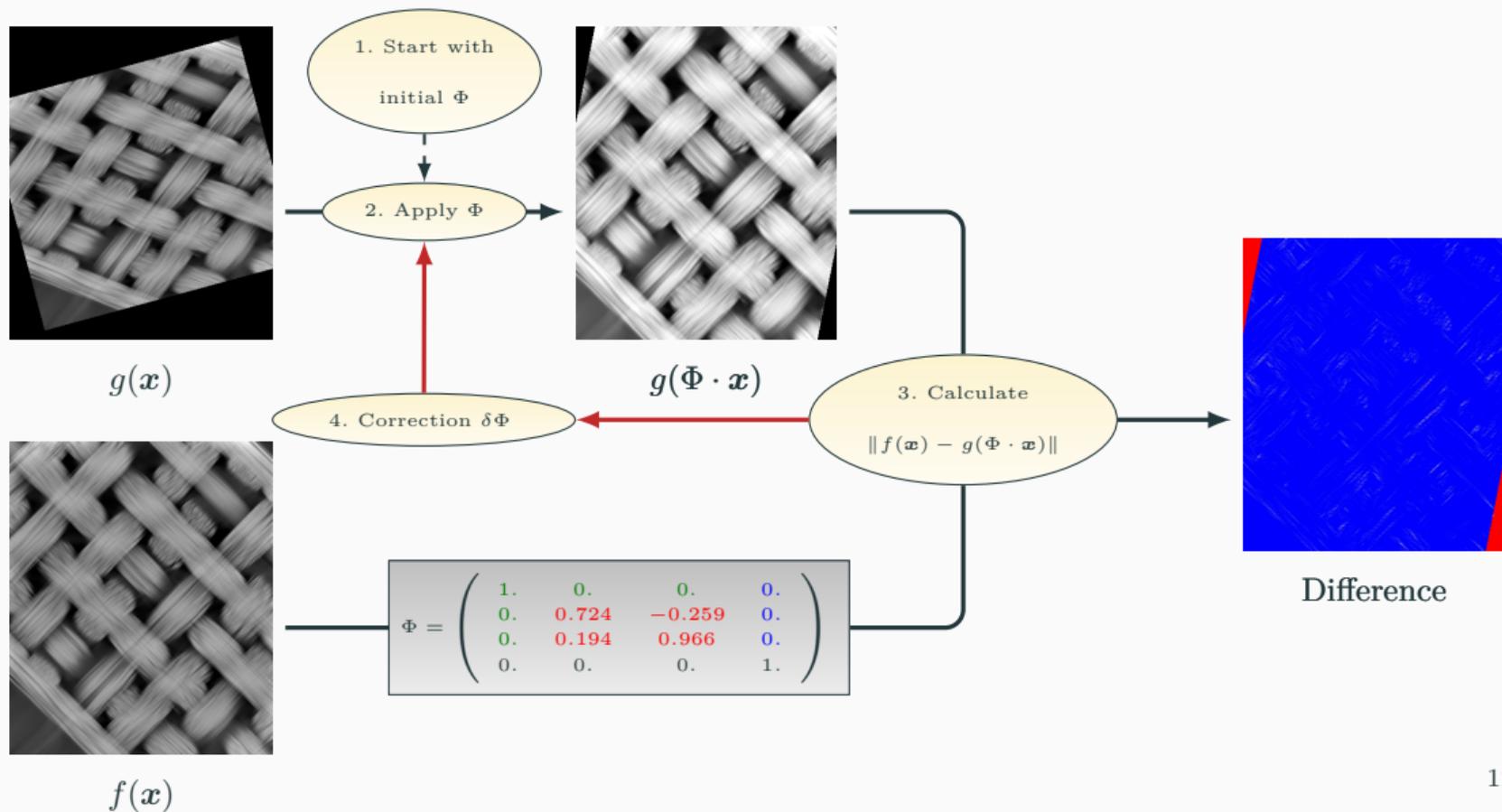


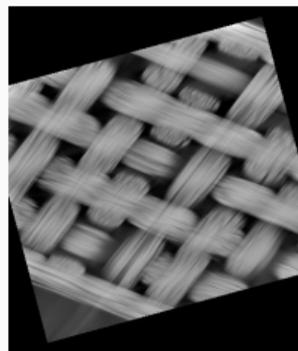




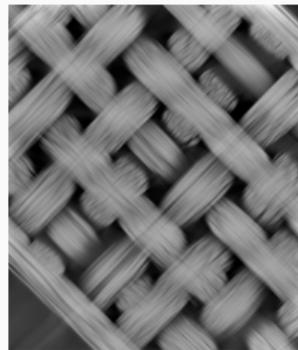




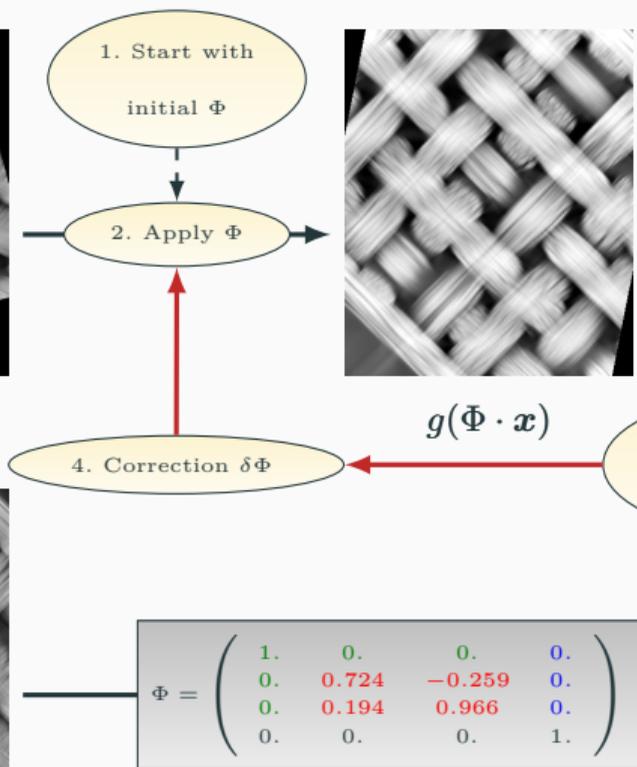




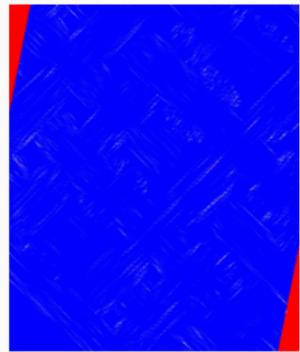
$g(x)$



$f(x)$



$$\mathcal{T}(\tilde{\Phi}) = \frac{1}{2} \sum_{\mathbf{x} \in \Omega} (f(\mathbf{x}) - g(\tilde{\Phi} \cdot \mathbf{x}))^2$$



Difference

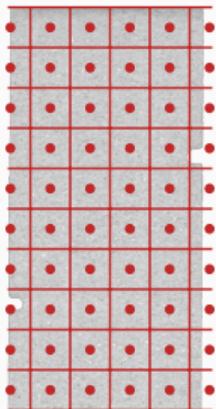
How to get a deformation **field**?

$$\mathcal{T}(\tilde{\Phi}) = \frac{1}{2} \sum_{\mathbf{x} \in \Omega} (f(\mathbf{x}) - g(\tilde{\Phi} \cdot \mathbf{x}))^2$$



How to get a deformation **field**?

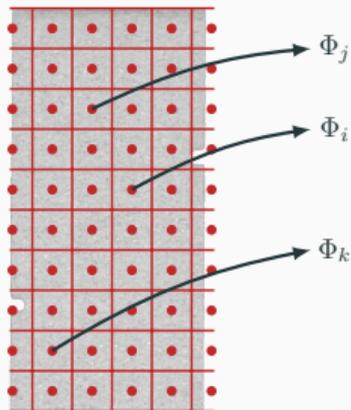
$$\mathcal{T}(\tilde{\Phi}) = \frac{1}{2} \sum_{\mathbf{x} \in \Omega} (f(\mathbf{x}) - g(\tilde{\Phi} \cdot \mathbf{x}))^2$$



Correlation windows
defined on a regular grid

How to get a deformation **field**?

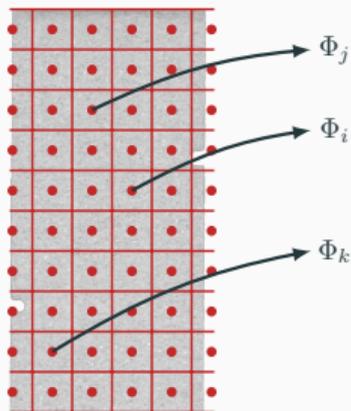
$$\mathcal{T}(\tilde{\Phi}) = \frac{1}{2} \sum_{\mathbf{x} \in \Omega} (f(\mathbf{x}) - g(\tilde{\Phi} \cdot \mathbf{x}))^2$$



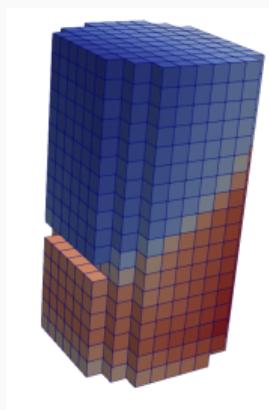
Correlation windows
defined on a regular grid

How to get a deformation **field**?

$$\mathcal{T}(\tilde{\Phi}) = \frac{1}{2} \sum_{\mathbf{x} \in \Omega} (f(\mathbf{x}) - g(\tilde{\Phi} \cdot \mathbf{x}))^2$$



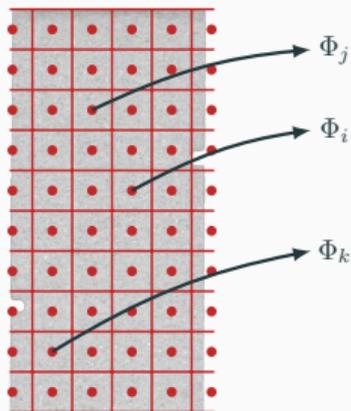
Correlation windows
defined on a regular grid



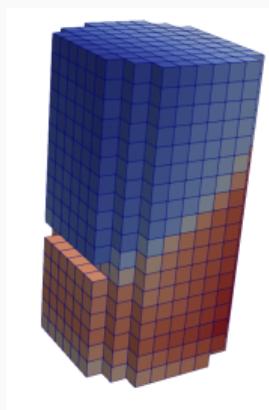
Displacement field

How to get a deformation field?

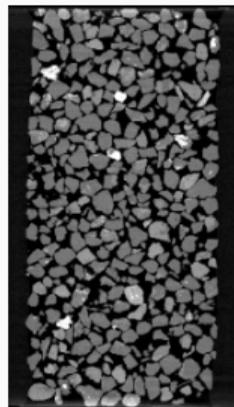
$$\mathcal{T}(\tilde{\Phi}) = \frac{1}{2} \sum_{\mathbf{x} \in \Omega} (f(\mathbf{x}) - g(\tilde{\Phi} \cdot \mathbf{x}))^2$$



Correlation windows
defined on a regular grid

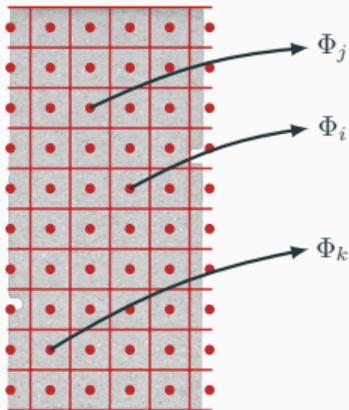


Displacement field

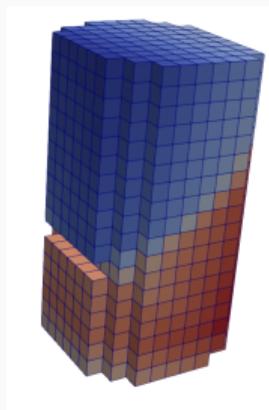


How to get a deformation field?

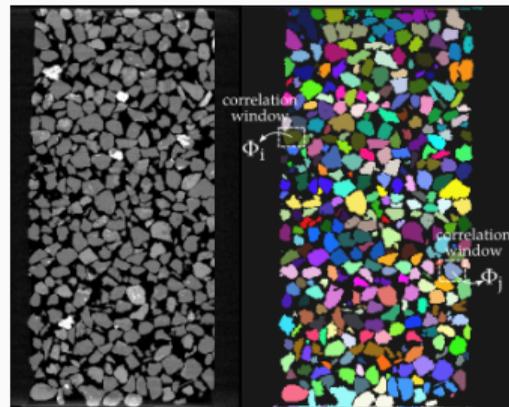
$$\mathcal{T}(\tilde{\Phi}) = \frac{1}{2} \sum_{\mathbf{x} \in \Omega} (f(\mathbf{x}) - g(\tilde{\Phi} \cdot \mathbf{x}))^2$$



Correlation windows
defined on a regular grid



Displacement field



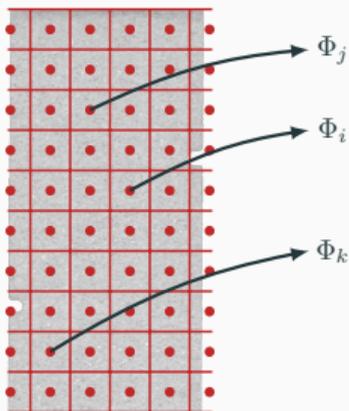
Correlation windows
defined on labels

 Hall S. et al.: *Discrete and continuum experimental study of localised deformation in hostun sand under triaxial compression using x-ray ct and 3d digital image correlation*, Géotechnique (2010)

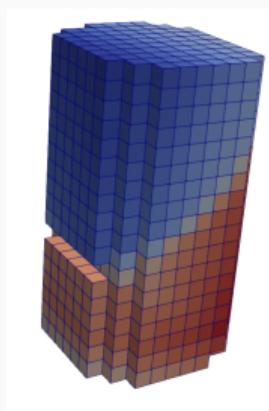
 Andô E. et al.: *Experimental micromechanics: grain-scale observation of sand deformation*, Géotechnique Letters (2012)

How to get a deformation field?

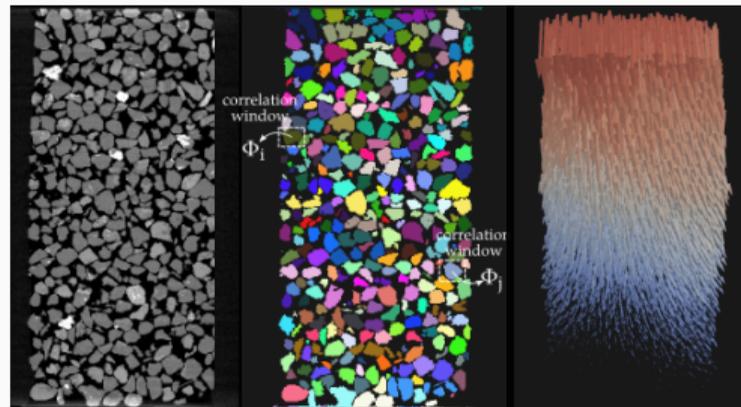
$$\mathcal{T}(\tilde{\Phi}) = \frac{1}{2} \sum_{\mathbf{x} \in \Omega} (f(\mathbf{x}) - g(\tilde{\Phi} \cdot \mathbf{x}))^2$$



Correlation windows
defined on a regular grid



Displacement field



Correlation windows
defined on labels

Displacement field

 Hall S. et al.: *Discrete and continuum experimental study of localised deformation in hostun sand under triaxial compression using x-ray ct and 3d digital image correlation*, Géotechnique (2010)

 Andô E. et al.: *Experimental micromechanics: grain-scale observation of sand deformation*, Géotechnique Letters (2012)

1. Initial registration script called from bash

```
1  bash> spam-register          # The script for the initial alingment
2  01.tif 02.tif                # The two tiff files to load
```

2. Regular grid DVC script called from bash

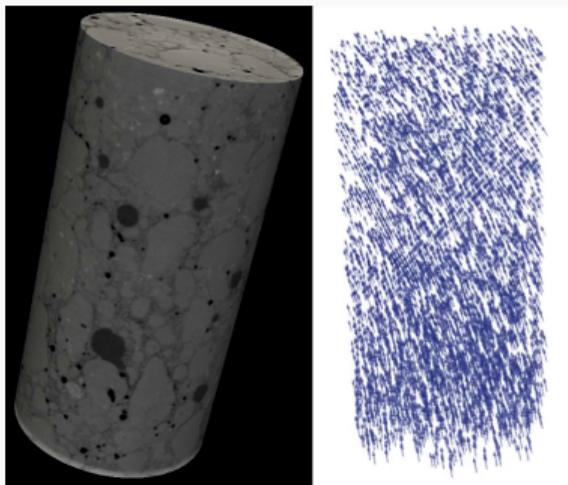
```
1  bash> spam-ldic             # The script for the structured grid
2  01.tif 02.tif \             # The two tiff files to load
3  -pf 01-02-registration.tsv \ # Initial guess from the script above
4  -hws X \                   # Half-size of the correlation window
5  -ns X \                    # The node spacing of the grid
6  -tsv -vtk -tif             # Ask for TSV, TIFF and VTK file outputs
```

3. Strains script called from bash

```
1  bash> spam-regularStrains   # The script for the structured strains
2  01-02-ldic.tsv \           # Deformation field from the script above
3  -tsv -vtk -tif             # Ask for TSV, TIFF and VTK file outputs
```

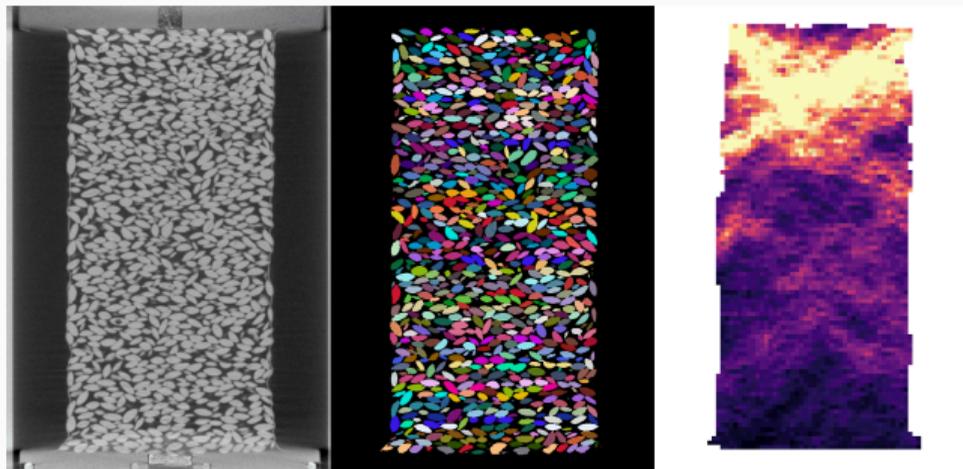
How to get a deformation **field** through **time**?

DVC on a regular grid



[Stamati O. (2021) Cem. Concr. Res.]

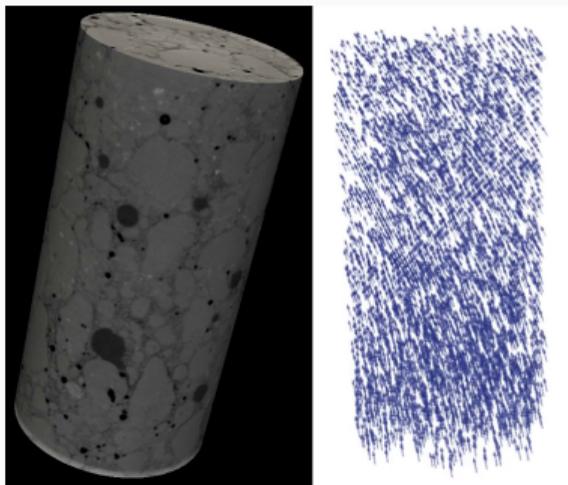
DVC on labelled objects



[Pinzón G. (2023) Gran. Matter]

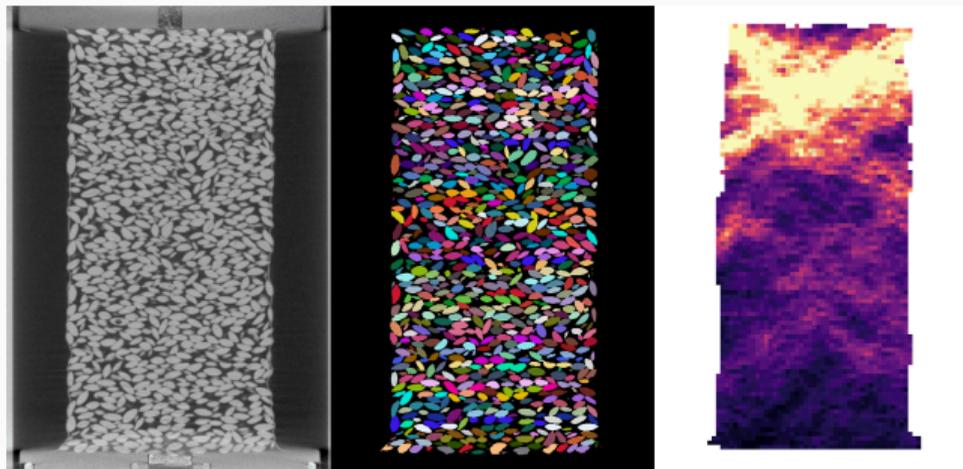
How to get a deformation **field** through **time**?

DVC on a regular grid



[Stamati O. (2021) Cem. Concr. Res.]

DVC on labelled objects

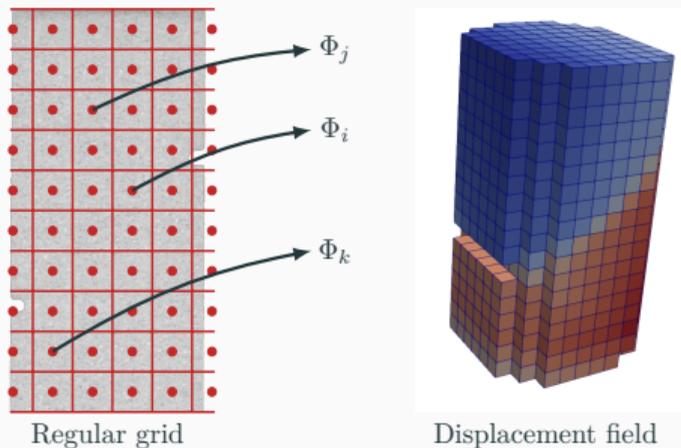


[Pinzón G. (2023) Gran. Matter]

Both are considered as **local** correlation approaches

Global vs Local DVC approaches

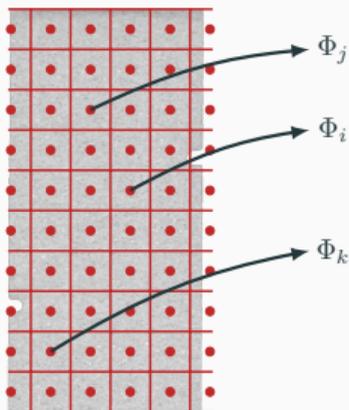
The paradigm behind the **global correlation** is to transform an image with a **displacement field** which support is a **Finite Element mesh** instead of a set of linear operators Φ .



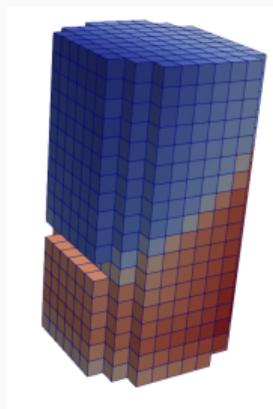
$$\mathcal{T}(\tilde{\Phi}) = \frac{1}{2} \sum_{\mathbf{x} \in \Omega} \left(f(\mathbf{x}) - g(\tilde{\Phi} \cdot \mathbf{x}) \right)^2$$

Global vs Local DVC approaches

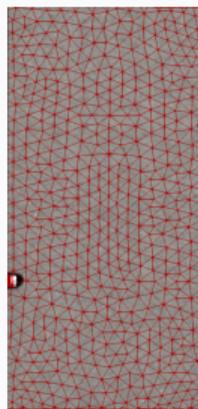
The paradigm behind the **global correlation** is to transform an image with a **displacement field** which support is a **Finite Element mesh** instead of a set of linear operators Φ .



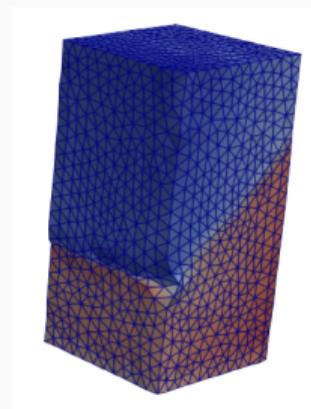
Regular grid



Displacement field



FE mesh



Displacement field

$$\mathcal{T}(\tilde{\Phi}) = \frac{1}{2} \sum_{\mathbf{x} \in \Omega} \left(f(\mathbf{x}) - g(\tilde{\Phi} \cdot \mathbf{x}) \right)^2$$

$$\Phi_{\mathbf{C}} = \sum_{\mathbf{x} \in \Omega} \frac{1}{2} \|g(\mathbf{x} + \mathbf{u}(\mathbf{x})) - f(\mathbf{x})\|_2$$

A Finite Element mesh to transform an image

The global DVC problem can be expressed as the following functional to minimize:

$$\Phi_c = \sum_{\Omega} \|g(\mathbf{x} + \mathbf{u}(\mathbf{x})) - f(\mathbf{x})\|_2$$

where:

- $\mathbf{u}(\mathbf{x}) = \mathbf{N}(\mathbf{x})\mathbf{d}$ is the displacement field supported but standard FE shape functions

A Finite Element mesh to transform an image

The global DVC problem can be expressed as the following functional to minimize:

$$\Phi_c = \sum_{\Omega} \|g(\mathbf{x} + \mathbf{u}(\mathbf{x})) - f(\mathbf{x})\|_2$$

In its weak form can be solved with a Newton-Raphson method:

$$\mathbf{M}_c \delta \mathbf{d} = \mathbf{b}$$

where:

- $\mathbf{u}(\mathbf{x}) = \mathbf{N}(\mathbf{x})\mathbf{d}$ is the displacement field supported by standard FE shape functions

A Finite Element mesh to transform an image

The global DVC problem can be expressed as the following functional to minimize:

$$\Phi_c = \sum_{\Omega} \|g(\mathbf{x} + \mathbf{u}(\mathbf{x})) - f(\mathbf{x})\|_2$$

In its weak form can be solved with a Newton-Raphson method:

$$\mathbf{M}_c \delta \mathbf{d} = \mathbf{b}$$

where:

- $\mathbf{u}(\mathbf{x}) = \mathbf{N}(\mathbf{x})\mathbf{d}$ is the displacement field supported by standard FE shape functions
- $\delta \mathbf{d}$ is the increment of displacement field unknowns \mathbf{d}
- \mathbf{M}_c can be seen as a FE assembled mass matrix
- \mathbf{b} contains the residual supported by the shape functions

Mechanically driven Tikhonov regularization

Equilibrium gap method: Minimizing the distance between current solution \mathbf{u} and that which satisfies the **equation for linear elasticity**.

$$\left\{ \begin{array}{l} \Phi_c(\mathbf{d}) = \sum_{\Omega} \|g(\mathbf{x} + \mathbf{u}(\mathbf{x})) - f(\mathbf{x})\|_2 \\ \end{array} \right.$$

The regularised problem becomes:

$$(\mathbf{M}_c + \mathbf{M}_{\text{reg}}) \delta \mathbf{d} = \mathbf{b} - \mathbf{M}_{\text{reg}} \mathbf{d}$$

Mechanical regularization

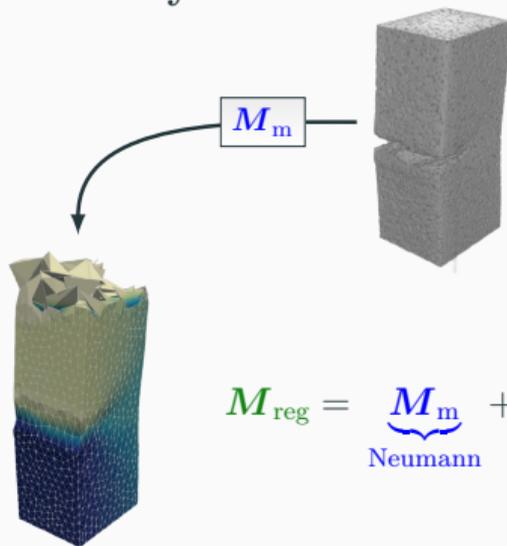
Mechanically driven Tikhonov regularization

Equilibrium gap method: Minimizing the distance between current solution \mathbf{u} and that which satisfies the **equation for linear elasticity**.

$$\left\{ \begin{array}{l} \Phi_c(\mathbf{d}) = \sum_{\Omega} \|g(\mathbf{x} + \mathbf{u}(\mathbf{x})) - f(\mathbf{x})\|_2 \\ \Phi_m(\mathbf{d}) = \|\mathbf{K}\mathbf{u} - \mathbf{f}\|_2 \quad (\text{Neumann}) \end{array} \right.$$

The regularised problem becomes:

$$(\mathbf{M}_c + \mathbf{M}_{\text{reg}}) \delta \mathbf{d} = \mathbf{b} - \mathbf{M}_{\text{reg}} \mathbf{d}$$



$$\mathbf{M}_{\text{reg}} = \underbrace{\mathbf{M}_m}_{\text{Neumann}} +$$



Mechanical regularization

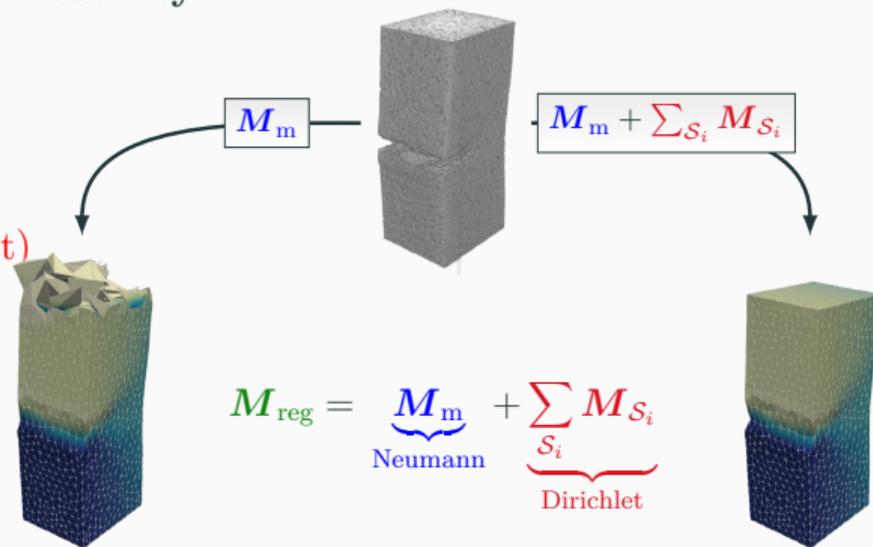
Mechanically driven Tikhonov regularization

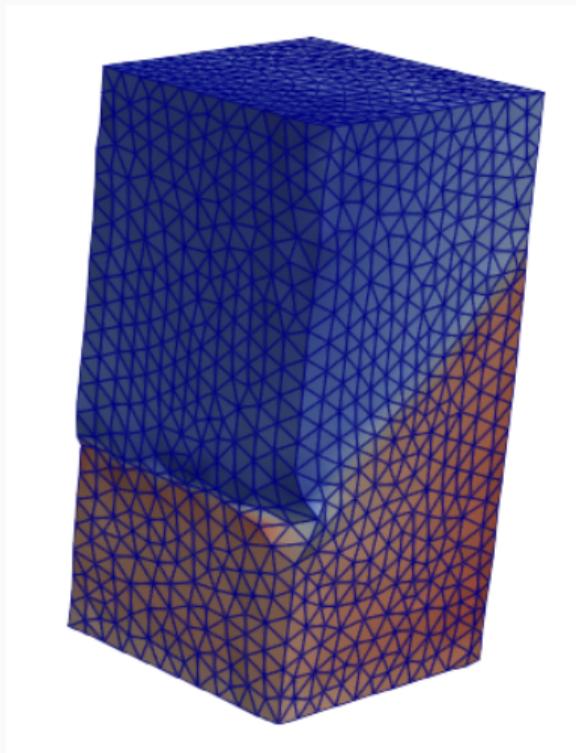
Equilibrium gap method: Minimizing the distance between current solution \mathbf{u} and that which satisfies the **equation for linear elasticity**.

$$\left\{ \begin{array}{l} \Phi_c(\mathbf{d}) = \sum_{\Omega} \|g(\mathbf{x} + \mathbf{u}(\mathbf{x})) - f(\mathbf{x})\|_2 \\ \Phi_m(\mathbf{d}) = \|\mathbf{K}\mathbf{u} - \mathbf{f}\|_2 \quad (\text{Neumann}) \\ \Phi_{S_i}(\mathbf{d}) = \int_{S_i} \|\nabla(\boldsymbol{\sigma} \cdot \mathbf{n})\|_2 d\mathbf{x} \quad (\text{Dirichlet}) \end{array} \right.$$

The regularised problem becomes:

$$(\mathbf{M}_c + \mathbf{M}_{\text{reg}}) \delta \mathbf{d} = \mathbf{b} - \mathbf{M}_{\text{reg}} \mathbf{d}$$





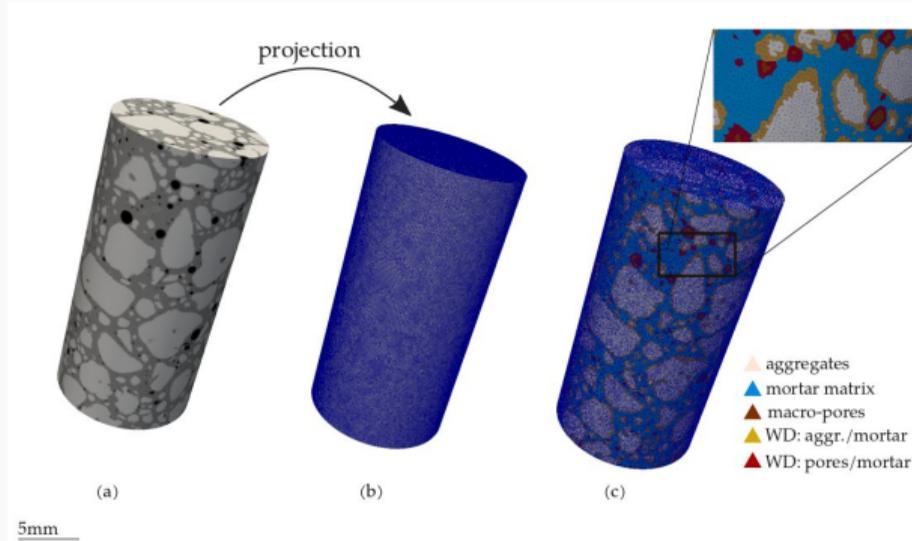
Advantages

- Continuity of sought displacement field
- Natural regularization
- Mechanically admissible solution
- Straightforward link with simulations
- Can work for non-elastic problems

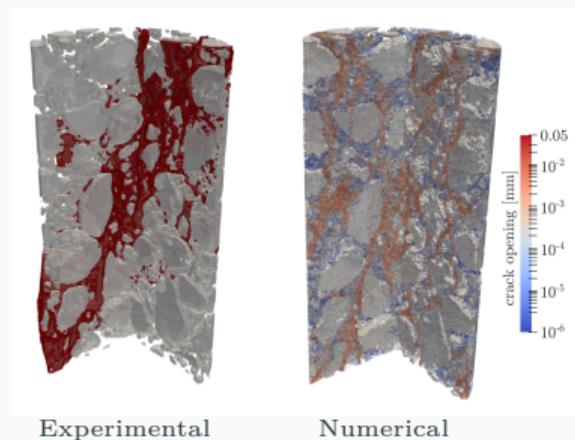
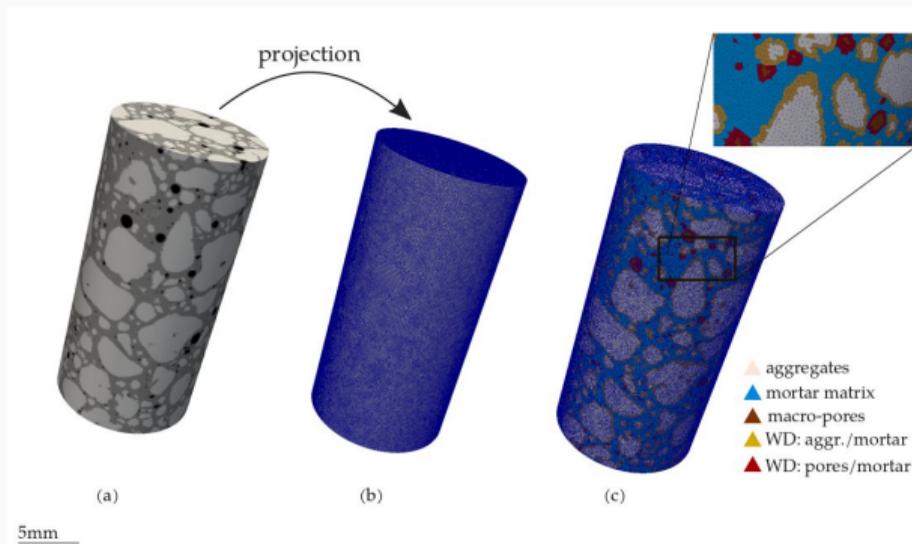
Global DVC in

- Recently implemented
- Looking for nice data to test it!

Bridging the gap between images and FE models



Bridging the gap between images and FE models

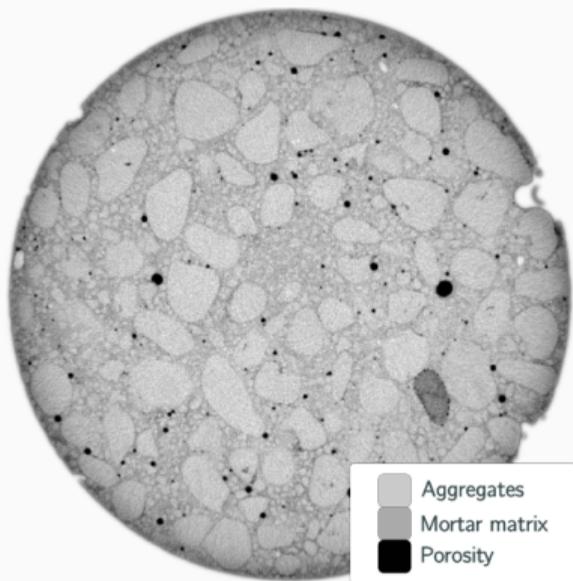


Stamati O., Roubin E., Andò E., Malecot Y.: *Tensile failure of micro-concrete: from mechanical tests to fe meso-model with the help of x-ray tomography*, Meccanica (2019)

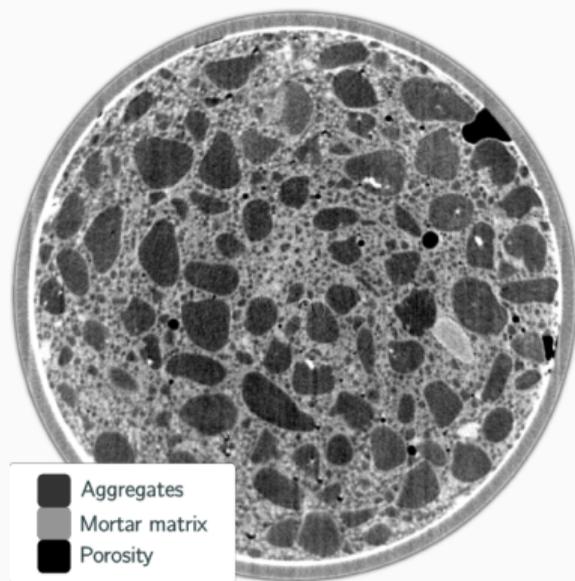


<https://ttk.gricad-pages.univ-grenoble-alpes.fr/spam/tutorials/tutorial-06-projection.html>

Multiple modalities

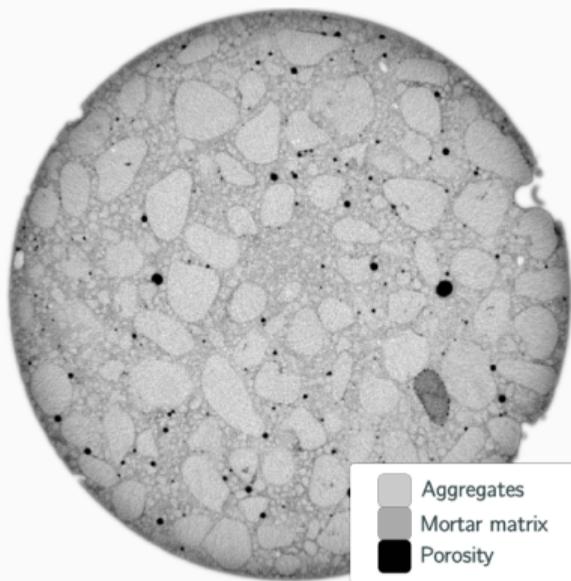


X-ray scan

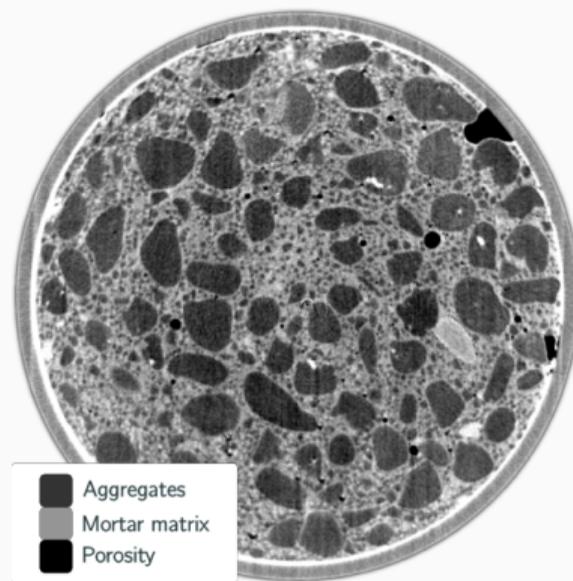


Neutrons scan from D50 (NeXT, ILL)

Multiple modalities



X-ray scan

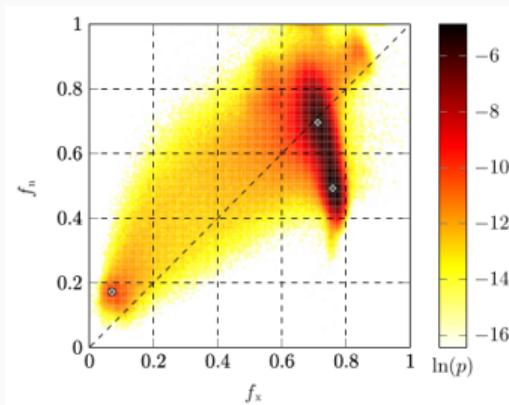


Neutrons scan from D50 (NeXT, ILL)

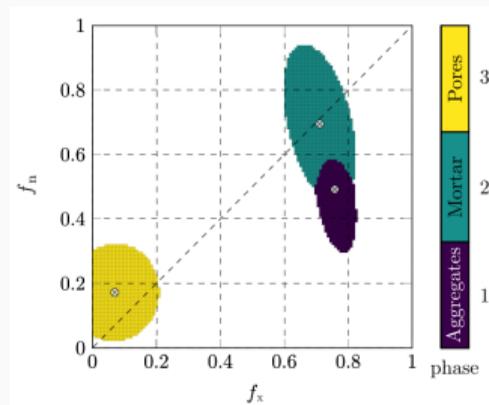
$$\mathcal{T}(\tilde{\Phi}) = \frac{1}{2} \sum_{x \in \Omega} (f(x) - g(\tilde{\Phi} \cdot x))^2 \quad ?$$

A solution

Construct a functional $\Phi(f_x, f_n)$ based the joint histogram $p(f_x, f_n)$.

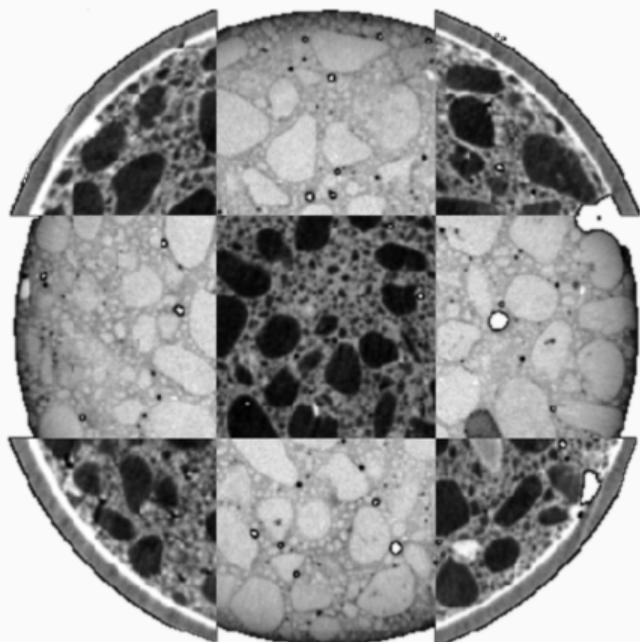


Joint histogram $p(f_x, f_n)$



Phase repartition

- Tudisco, E., Jallin C., et al.: *An extension of digital volume correlation for multimodality image registration*, Measurement Science and Technology (2017)
- Roubin E., Andò E., Roux S.: *The colours of concrete as seen by x-rays and neutrons*, Cement and Concrete Composites (2019)



MMR in **spam** tomical analysis

- Online tutorials and examples
- Used in many published works

Going further

- Enhance spatial resolution of neutrons
- Kinematics that map one phase to the next image

General information

spam 0.7.1.0

 Latest version

```
pip install spam
```

Released: Mar 6, 2024

Software for the Practical Analysis of Materials

Navigation

 Project description

 Release history

 Download files

Verified details

These details have been verified by PyPI

Maintainers



edwardando



eroubin



gustavoPinzon



ostamati



remche

Project description

[license](#) [GPLv3](#)
[pipeline](#) [spaced](#)
[coverage](#) [92.50%](#)
[pypi package](#) [0.7.1.0](#)
[JOSS](#) [10.21105/joss.02286](#)

[downloads/month](#) [4k](#)
[PyPI Chat](#)
[join](#)

Spam is a piece of Python software built upon NumPy and SciPy for the analysis and manipulation of 3D and 2D data sets in material science, be they from x-ray tomography, random fields or any other source.

A number of common functions are provided that are either lacking or slow in NumPy and SciPy, which are expected to be used by users within new python scripts. These functions are in the `tools/` directory, and include tools to work with random fields, morphological operations, digital image correlation, and labelled images. Some of spam's functions transparently call C/C++ functions for speed.

Some user-callable scripts are also provided - they are more complex pieces of code that combine a number of functions and which have a command-line interface. For the moment the scripts are 3 different image correlation techniques.

Please have a look at our online documentation for:

- [Installation instructions](#)
- [General introduction](#)
- [Examples](#)
- And a number of detailed tutorials

If you find bugs, need help, or want to talk to the developers, we use a [element.io/matrix.org](#) chat room for organisation, please join it [here](#) and come and talk to us - it is easy, there is a chat client that can run in your web browser. All you need to do is choose a user name!

downloads 290k

downloads/month 4k

downloads/week 647



Stamati, Andô, Roubin: *Journal of Open Source Software*, 2020 (Citations: 110)


<https://pepy.tech/project/spam>

spam 0.7.1.0

 Latest version

```
pip install spam
```

Released: Mar 6, 2024

Software for the Practical Analysis of Materials

Navigation

 Project description

 Release history

 Download files

Verified details

These details have been verified by PyPI

Maintainers



edwardando



eroubin



gustavoPinzon



ostamati



remche

Project description

license	GPLv3	pipeline	spaced	coverage	92.50%	pypi package	0.7.1.0	JOSS	10.21105/joss.02286
downloads/month	4k	PyPI Chat	join						

Spam is a piece of Python software built upon NumPy and SciPy for the analysis and manipulation of 3D and 2D data sets in material science, be they from x-ray tomography, random fields or any other source.

A number of common functions are provided that are either lacking or slow in NumPy and SciPy, which are expected to be used by users within new python scripts. These functions are in the `tools/` directory, and include tools to work with random fields, morphological operations, digital image correlation, and labelled images. Some of spam's functions transparently call C/C++ functions for speed.

Some user-callable scripts are also provided - they are more complex pieces of code that combine a number of functions and which have a command-line interface. For the moment the scripts are 3 different image correlation techniques.

Please have a look at our online documentation for:

- [Installation instructions](#)
- [General introduction](#)
- [Examples](#)
- And a number of detailed tutorials

If you find bugs, need help, or want to talk to the developers, we use a element.io/matrix.org chat room for organisation, please join it [here](#) and come and talk to us - it is easy, there is a chat client that can run in your web browser. All you need to do is choose a user name!

downloads 290k

downloads/month 4k

downloads/week 647

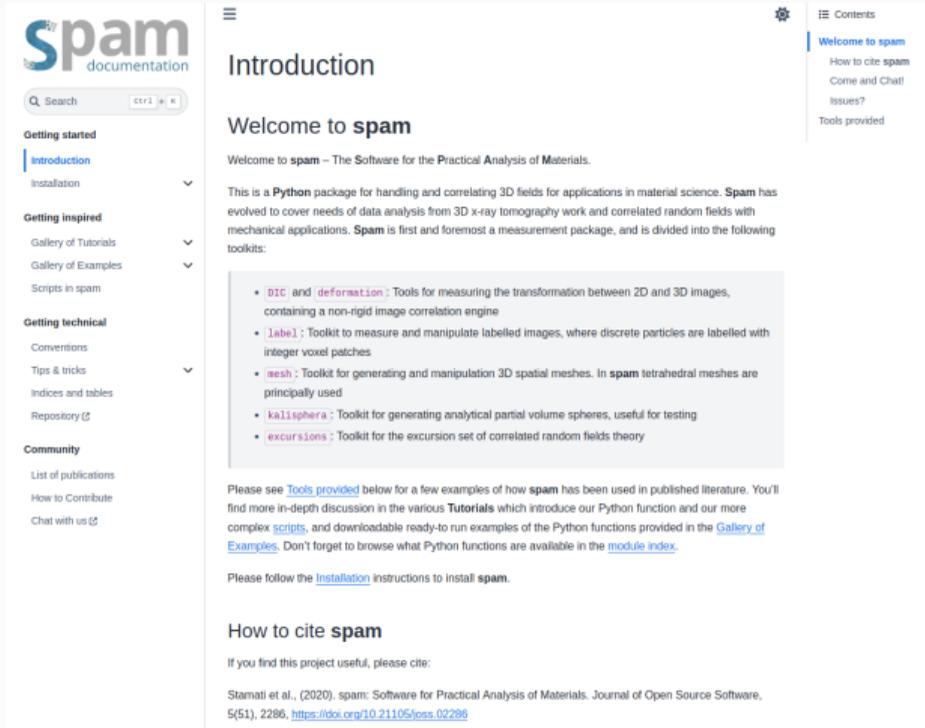


Stamati, Andô, Roubin: *Journal of Open Source Software*, 2020 (Citations: 110)

Easy to use!!

```
pip install spam
```


<https://pepy.tech/project/spam>



The screenshot shows the 'spam' documentation website. The header includes the 'spam' logo and 'documentation' text. A search bar is present with a 'ctrl k' shortcut. The left sidebar contains navigation links: 'Getting started' (with 'Introduction' selected), 'Getting inspired', and 'Getting technical'. The main content area is titled 'Introduction' and 'Welcome to spam'. It contains a welcome message, a list of toolkits (DIC and deformation, label, mesh, kallispera, excursions), a section on how to cite 'spam', and a 'Read the Docs' icon in the bottom left corner.

spam documentation

Q Search

Getting started

- Introduction
- Installation

Getting inspired

- Gallery of Tutorials
- Gallery of Examples
- Scripts in spam

Getting technical

- Conventions
- Tips & tricks
- Indices and tables
- Repository

Community

- List of publications
- How to Contribute
- Chat with us

Introduction

Welcome to spam

Welcome to **spam** – The Software for the Practical Analysis of Materials.

This is a **Python** package for handling and correlating 3D fields for applications in material science. **Spam** has evolved to cover needs of data analysis from 3D x-ray tomography work and correlated random fields with mechanical applications. **Spam** is first and foremost a measurement package, and is divided into the following toolkits:

- **DIC and deformation**: Tools for measuring the transformation between 2D and 3D images, containing a non-rigid image correlation engine
- **label**: Toolkit to measure and manipulate labelled images, where discrete particles are labelled with integer voxel patches
- **mesh**: Toolkit for generating and manipulation 3D spatial meshes. In **spam** tetrahedral meshes are principally used
- **kallispera**: Toolkit for generating analytical partial volume spheres, useful for testing
- **excursions**: Toolkit for the excursion set of correlated random fields theory

Please see [Tools provided](#) below for a few examples of how **spam** has been used in published literature. You'll find more in-depth discussion in the various **Tutorials** which introduce our Python function and our more complex [scripts](#), and downloadable ready-to-run examples of the Python functions provided in the [Gallery of Examples](#). Don't forget to browse what Python functions are available in the [module index](#).

Please follow the [Installation](#) instructions to install **spam**.

How to cite spam

If you find this project useful, please cite:

Stamati et al., (2020). spam: Software for Practical Analysis of Materials. Journal of Open Source Software, 5(51), 2286, <https://doi.org/10.21105/joss.02286>

Read the Docs

- Installation instructions
- Tutorials
- Examples
- Module indices
- Code repository
- Communication
- How to contribute

The screenshot shows the GitLab interface for the 'spam' repository. The main content area displays a list of files and their commit history:

Name	Last commit	Last update
docs	how to contribute workflow	3 weeks ago
examples	[skip-ci] // examples / doc	2 months ago
src/spam	option to save deformed image in spam...	3 weeks ago
tests	DIC tests	2 months ago
.gitignore	pre commit	2 months ago
.gitlab-ci.yml	[skip-ci] // examples / doc	2 months ago
jsort.cfg	remove _script from gui scripts	1 month ago
pre-commit-config.yml	install makefile	1 year ago
LICENSE.md	small fixes to strains + beginning of GP...	4 years ago
MANIFEST.in	Will it pass the tests?	1 year ago
README.txt	prepare 0.71.0	1 month ago
build-wheels-OSX.sh	making the OSX build system more rob...	3 months ago
build-wheels.sh	Ready for 0.6.5.1	4 months ago
makefile	[skip-ci] // examples / doc	2 months ago
pyproject.toml	remove _script from gui scripts	1 month ago
setup.py	all scripts in new format and out of scri...	2 months ago

Project information on the right side includes: 2,377 Commits, 60 Branches, 50 Tags, 3 Releases, README, GNU GPLv3, and GitLab Pages. It was created on April 11, 2023.

- Source code
- 93% test coverage
- Reporting issues
- Contributing!



<https://gitlab.com/spam-project/spam>

Valais, Switzerland 18-21/07/2023



<https://www.spam-project.dev/workshops/2023/07/>

Villar d'Arène, France 6-9/10/2024



<https://www.spam-project.dev/workshops/2024/10/>

Valais, Switzerland 18-21/07/2023



<https://www.spam-project.dev/workshops/2023/07/>

Villar d'Arène, France 6-9/10/2024



<https://www.spam-project.dev/workshops/2024/10/>

Registrations are open !

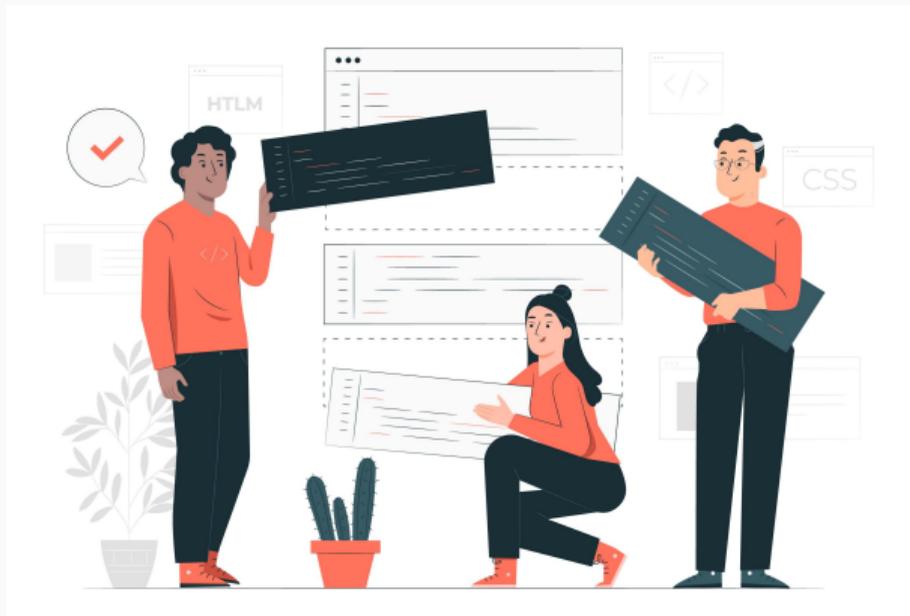
People involved



- Core-developers



People involved



- Core-developers



- Contributed with code



People involved



- Core-developers



- Contributed with code



- Contributed with ideas

