spam
ractical analysis

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

Outline

What is 

?

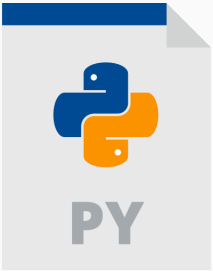
Let's play with some images

General information

What is  ?

What is Spam ?

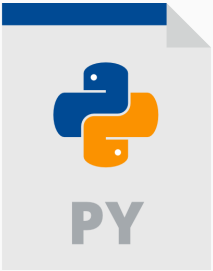
`python` library



(with some bits of `C++`)

What is Spam (practical analysis) ?

`python` library



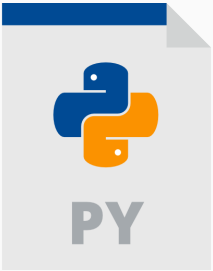
(with some bits of `C++`)

data analysis

(not AI enhanced!)

What is Spam (practical analysis) ?

`python` library

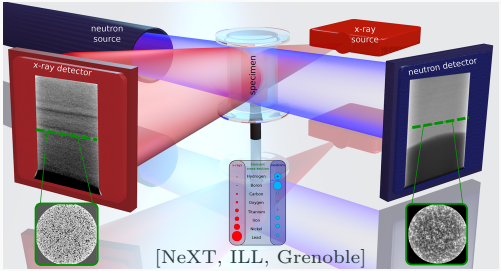


(with some bits of `C++`)

data analysis

(not AI enhanced!)

2D images and 3D volumes

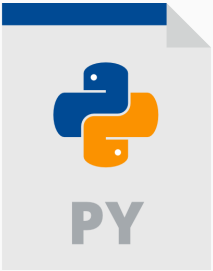


[NeXT, ILL, Grenoble]

X-ray & Neutron tomography
(for example)

What is Spam (practical analysis) ?

`python` library

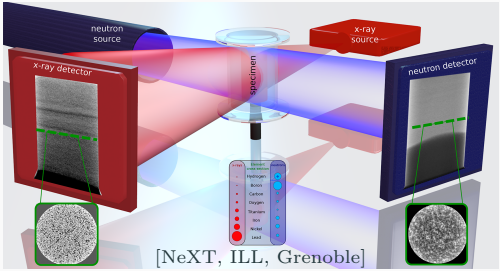


(with some bits of `C++`)

data analysis

(not AI enhanced!)

2D images and 3D volumes

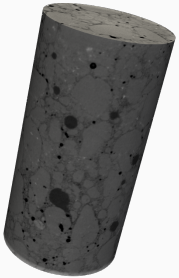


X-ray & Neutron tomography
(for example)

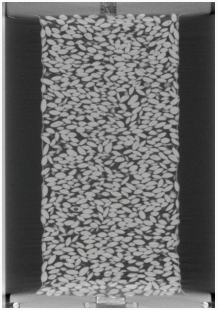
Strictly open-source !

Why do we need a software like Spam ?

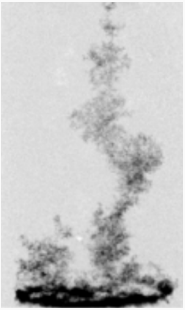
Concrete



Lentils

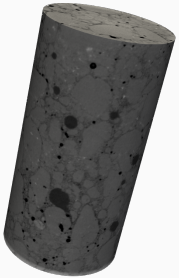


Sandstone

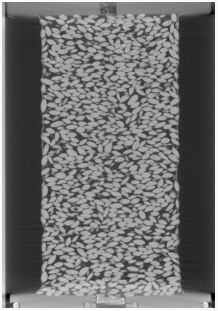


Why do we need a software like Spam ?

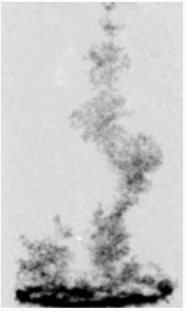
Concrete



Lentils



Sandstone

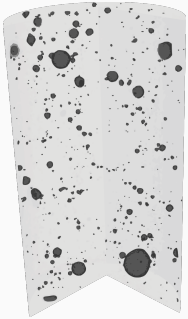


MORPHOLOGY

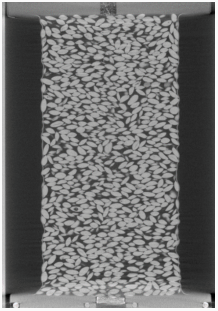
- Phase identification
- Volume fractions
- Size, shape
- Orientation, fabric
- Tortuosity, connectivity

Why do we need a software like Spam ractical analysis ?

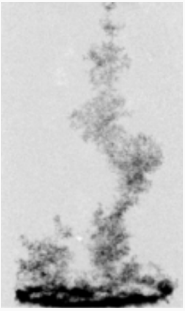
Concrete



Lentils



Sandstone

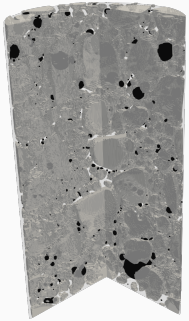


MORPHOLOGY

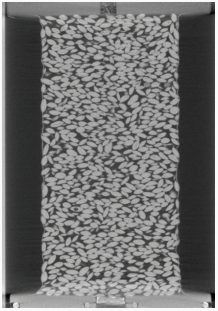
- Phase identification
- Volume fractions
- Size, shape
- Orientation, fabric
- Tortuosity, connectivity

Why do we need a software like Spam ?

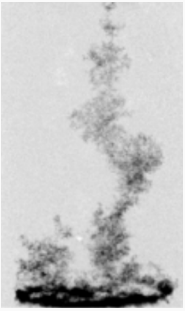
Concrete



Lentils



Sandstone

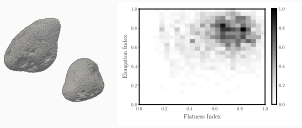


MORPHOLOGY

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

Why do we need a software like Spam ractical analysis ?

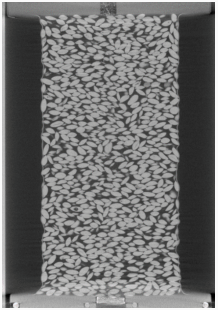
Concrete



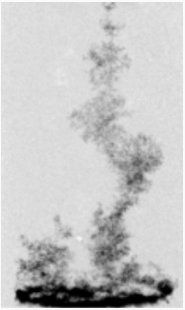
(a) aggregates

(b) macro-pores

Lentils



Sandstone

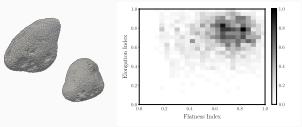


MORPHOLOGY

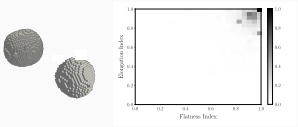
- Phase identification
- Volume fractions
- Size, shape
- Orientation, fabric
- Tortuosity, connectivity

Why do we need a software like Spam ?

Concrete

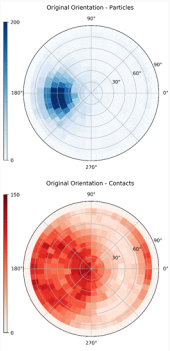


(a) aggregates

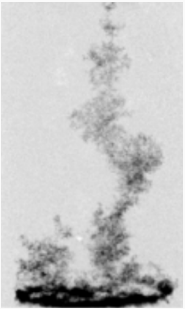


(b) macro-pores

Lentils



Sandstone

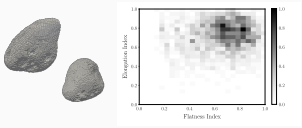


MORPHOLOGY

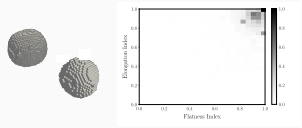
- Phase identification
- Volume fractions
- Size, shape
- Orientation, fabric
- Tortuosity, connectivity

Why do we need a software like Spam practical analysis ?

Concrete

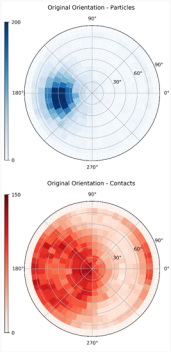


(a) aggregates

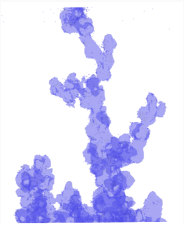


(b) macro-pores

Lentils



Sandstone

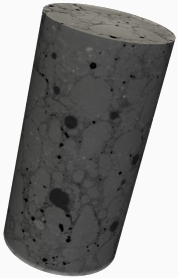


MORPHOLOGY

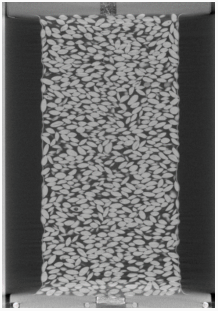
- Phase identification
- Volume fractions
- Size, shape
- Orientation, fabric
- Tortuosity, connectivity

Why do we need a software like Spam practical analysis ?

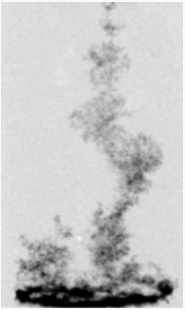
Concrete



Lentils



Sandstone

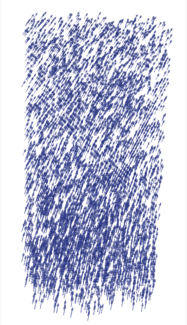


MORPHOLOGICAL EVOLUTION

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

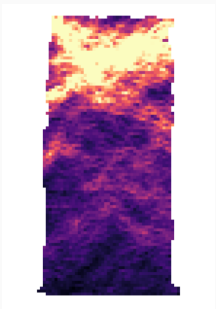
Why do we need a software like Spam practical analysis ?

Concrete displacement field



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

Lentils strain field



[Pinzón G. (2023) Gran. Matter]

Flow in sandstone

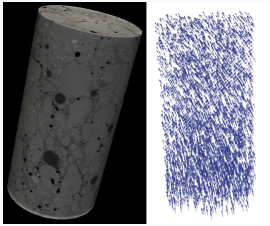
[Couture C. (in preparation)]

MORPHOLOGICAL EVOLUTION

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

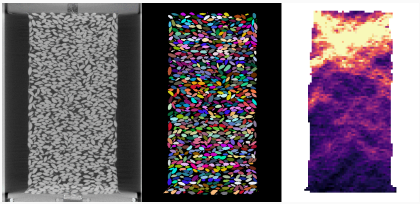
Examples of measuring deforming media with Spam

Concrete fracture



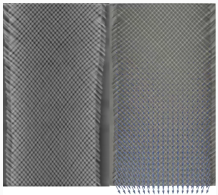
[Stamati O. (2021) Cem. Concr. Res.]

Particle tracking



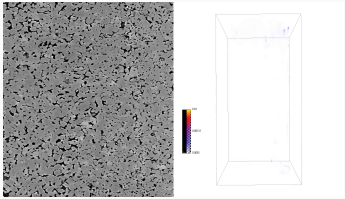
[Pinzón G. (2023) Gran. Matter]

Textile



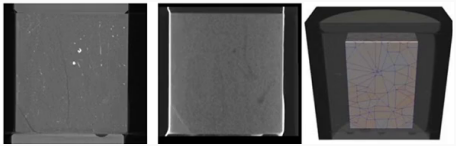
[Stamati O. (2023) Comp. Part A]

Sandstone failure



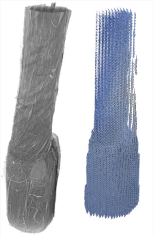
[Cartwright-Taylor A. (2022) Nature Comm.]

Claystone water absorption



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

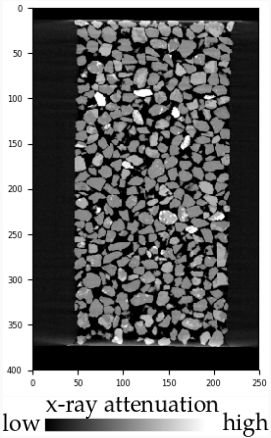
Tendon/Bone Interface



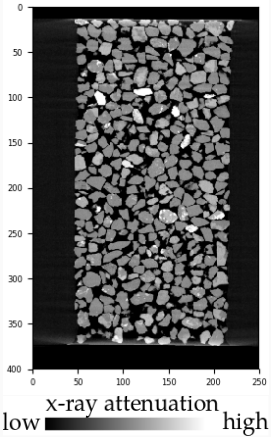
[Sensini A. (in preparation)]

Let's play with some images

Inside an X-ray image

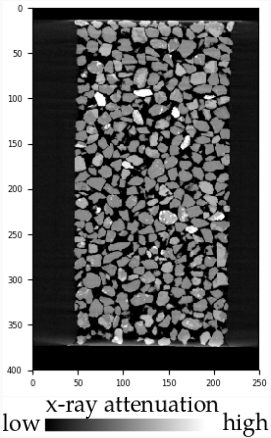


Inside an X-ray image

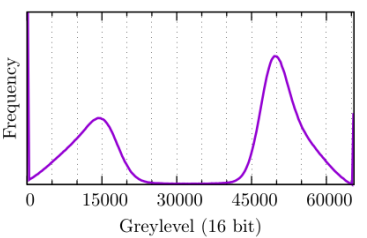



```
array([[33171, 29859, 31265, 28414, 30618, 31769,     0,     0,     0,
        16241, 29112, 29486],
       [28666, 27491, 31498, 28393, 31080, 22877,     0,   775,    43,
           34, 27448, 28065],
       [31150, 33212, 32147, 29513, 28288,   761,   261,     0,     0,
            2,  7497, 28411],
       [32261, 29453, 30281, 13581,   137,     0,    10,     0,     0,
           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)
```

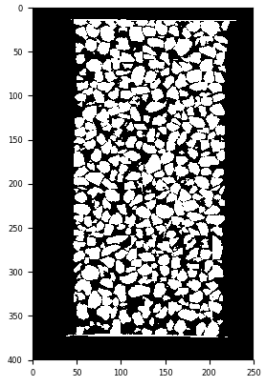
Inside an X-ray image




```
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)
```

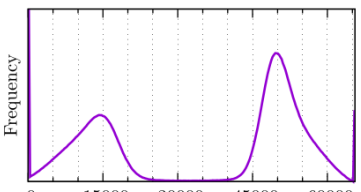


Inside an X-ray image

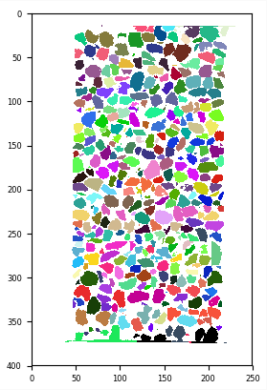


Binarised image

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

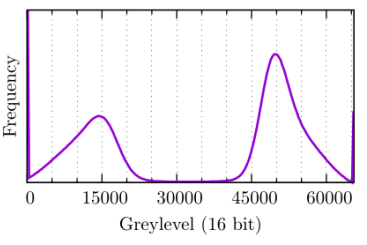


Inside an X-ray image

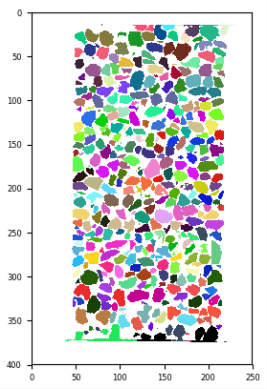


Labelled image

```
[[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]]
```

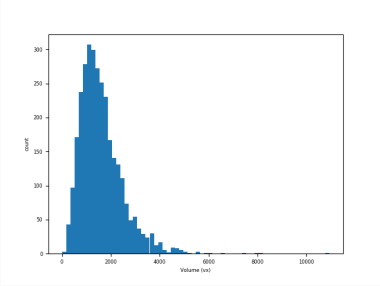


Inside an X-ray image

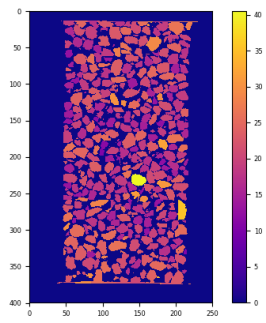


Labelled image

```
[[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 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]]
```

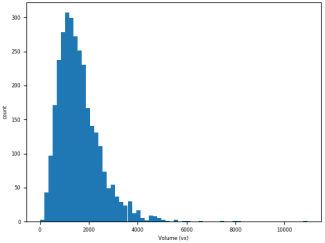


Inside an X-ray image

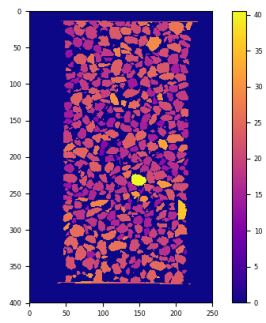


Particle volumes image

```
[[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 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]]
```

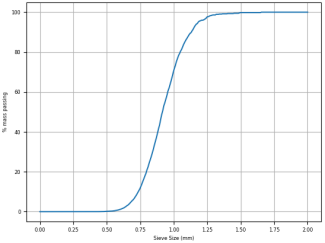


Inside an X-ray image



Particle volumes image

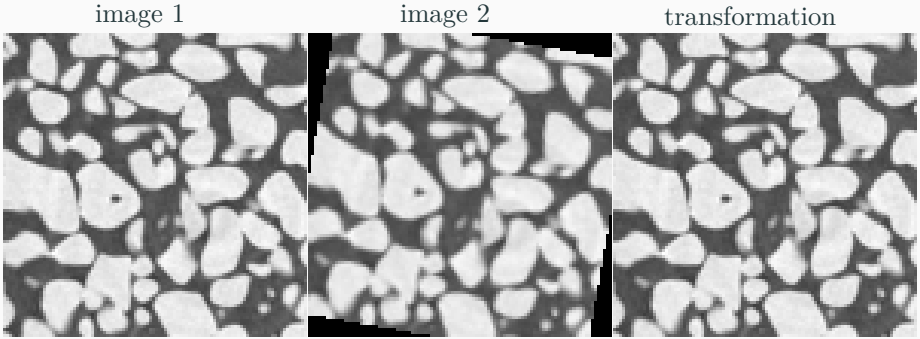
```
[[1583 1583 1583 1583 1583 1583    0    0    0    0 1614 1614]
 [1583 1583 1583 1583 1583 1583    0    0    0    0 1614 1614]
 [1583 1583 1583 1583 1583    0    0    0    0    0    0 1614]
 [1583 1583 1583    0    0    0    0    0    0    0    0 1614]
 [1583    0    0    0    0    0    0    0   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



example: Morphological measurements

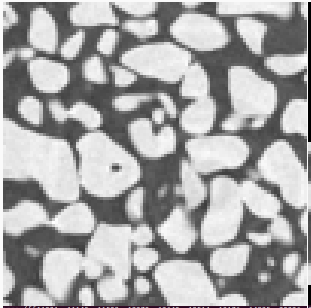
```python
# import modules
import numpy, tifffile
import spam.label, spam.plotting

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

What about the evolution through time?



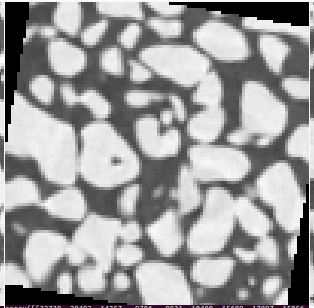
What about the evolution through time?

image 1



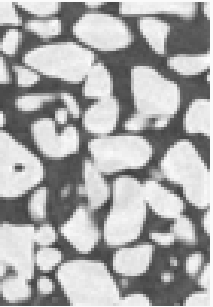
```
array([[11931, 11145, 11508, 12270, 12640, 12514, 13176, 14835, 20326,
        29225, 34191, 21655, 12285],
       [18759, 12223, 12034, 12864, 13021, 12939, 12926, 16789, 23822,
        29928, 33987, 26894, 13810],
       [27971, 15812, 12261, 12486, 13352, 12963, 12321, 17837, 25474,
        31316, 33726, 31330, 17174],
       [30987, 18643, 12490, 13015, 13259, 12608, 12316, 17200, 26466,
        31433, 32565, 32627, 24529],
       [30940, 20799, 13864, 13810, 12368, 12428, 12638, 14913, 22127,
        28696, 30402, 31378, 29082],
       [30388, 20819, 13442, 13171, 12174, 11788, 13367, 15867, 18360,
        20486, 23084, 24087, 21227],
       [28238, 16996, 12774, 11774, 10907, 11451, 17046, 24189, 26307,
        18931, 12581, 12947, 11612],
       [20842, 11917, 12057, 11458, 10142, 12744, 21917, 30117, 32083,
        26414, 13983, 9227, 9560],
       [11830, 10643, 11377, 11926,  9810, 12976, 24110, 30924, 32738,
        28492, 14357,  9786,  9831],
       [10034, 10527, 11255, 11807, 10928, 11390, 18715, 25353, 27755,
        21958, 12138, 10322, 10209],
       [ 9231,  9982, 11295, 11915, 11800, 11152, 12225, 15596, 16983,
        12342, 10224, 10482, 11079],
       [10336,  9713, 11265, 12484, 11966, 11322, 11061, 10848, 10408,
         9832,  9688, 11002, 11866],
       [13713, 11569, 12586, 13535, 11586, 10646, 10546, 10882, 10976,
        10456,  9832,  9996, 10933]], dtype=uint16)
```

image 2



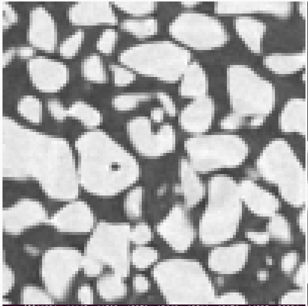
```
array([[32738, 28492, 14357,  9786,  9831, 10480, 15689, 17897, 19056,
        11978, 11473, 11169, 11038],
       [27755, 21958, 12138, 10322, 10209,  9708, 12119, 16130, 15933,
        13200, 12479, 11724, 11014],
       [16983, 12342, 10224, 10482, 11079, 10295, 11008, 14399, 16311,
        14135, 12619, 11464, 10691],
       [10408,  9832,  9688, 11002, 11866, 10993, 11439, 12952, 15983,
        16938, 19381, 21616, 19468],
       [10976, 10456,  9832,  9996, 10933, 11927, 11476, 11817, 13966,
        20464, 28899, 30485, 28835],
       [11106, 10810, 10411, 10206, 10408, 10965, 11633, 12334, 12732,
        19483, 29714, 31859, 30801],
       [17810, 11712, 10137, 10624, 11086, 10827, 11452, 11589, 11653,
        16397, 26023, 29611, 26193],
       [32045, 26023, 17541, 11582, 11310, 12237, 12162, 12085, 11463,
        12752, 18506, 26914, 16539],
       [33943, 33966, 32197, 21946, 11953, 13484, 13231, 13101, 12433,
        12164, 13428, 13824, 12163],
       [33859, 32898, 34371, 32940, 16905, 12897, 13213, 12454, 12711,
        12339, 12456, 13448, 13074],
       [33859, 32843, 32587, 34144, 24247, 12734, 12364, 11712, 11527,
        12905, 13632, 13041, 12610],
       [32681, 32889, 32693, 34115, 25019, 12476, 12798, 10718, 11111,
        12560, 12814, 12958, 11956],
       [26569, 32452, 33644, 31245, 17666, 11383, 12677, 11976, 11876,
        12189, 12612, 12771, 11369]], dtype=uint16)
```

transformation



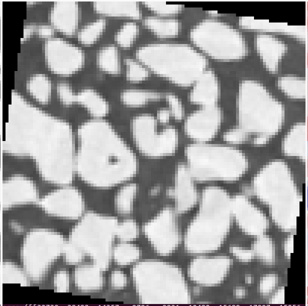
What about the evolution through time?

image 1



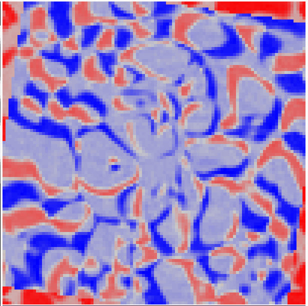
```
array([[11931, 11145, 11508, 12270, 12640, 12514, 13176, 14835, 20326,
        29225, 34191, 21655, 12285],
       [18759, 12223, 12034, 12864, 13021, 12939, 12926, 16789, 23822,
        29928, 33987, 26894, 13810],
       [27971, 15814, 12261, 12486, 13352, 12963, 12321, 17037, 25474,
        31316, 33720, 31330, 17174],
       [30987, 18643, 12490, 13615, 13259, 12608, 12316, 17200, 26466,
        31433, 32565, 32627, 24529],
       [30940, 20799, 13864, 13810, 12368, 12428, 12638, 14913, 22127,
        28696, 30402, 31378, 29082],
       [30308, 20819, 13442, 13171, 12174, 11788, 13367, 15867, 18360,
        20400, 23084, 24087, 21327],
       [28231, 16996, 12774, 11774, 10907, 11451, 17046, 24109, 26307,
        18931, 12581, 12947, 11612],
       [20042, 11917, 12057, 11458, 10142, 12744, 21917, 30117, 32083,
        26414, 13083,  9227,  9560],
       [11830, 10643, 11377, 11326,  9810, 12978, 24110, 30924, 32738,
        28492, 14357,  9786,  9831],
       [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, 11866],
       [13713, 11569, 12586, 13535, 11586, 10646, 10546, 10882, 10976,
        10456,  9832,  9996, 10933]], dtype=uint16)
```

image 2



```
array([[32738, 28492, 14357,  9786,  9831, 10480, 15689, 17897, 15056,
        11978, 11473, 11169, 11038],
       [27755, 21958, 12138, 10322, 10209,  9708, 12119, 16130, 15933,
        13200, 12479, 11724, 11014],
       [16983, 12342, 10224, 10482, 11079, 10295, 11008, 14399, 16311,
        14135, 12619, 11464, 10691],
       [10408,  9032,  9688, 11002, 11866, 10993, 11439, 12952, 15983,
        16938, 19381, 21616, 19468],
       [10976, 10456,  9832,  9996, 10933, 11927, 11476, 11817, 13966,
        20464, 28899, 30485, 28835],
       [11100, 10816, 10411, 10206, 10400, 10965, 11633, 12334, 12732,
        19483, 29714, 31859, 30801],
       [17810, 11712, 10137, 10624, 11086, 10827, 11452, 11589, 11653,
        16397, 26023, 29611, 26193],
       [32045, 26023, 17541, 11582, 11310, 12237, 12162, 12005, 11463,
        12752, 18506, 20914, 16539],
       [33943, 33966, 32197, 21946, 11953, 13404, 13231, 13101, 12433,
        12164, 13428, 13824, 12163],
       [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],
       [32681, 32889, 32693, 34115, 25019, 12476, 12798, 10718, 11111,
        12560, 12814, 12958, 11956],
       [26569, 32452, 33644, 31245, 17666, 11383, 12677, 11976, 11876,
        12189, 12612, 12771, 11369]], dtype=uint16)
```

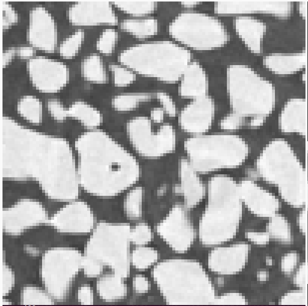
residual



```
array([[20807, 17347,  2849, 63052, 62727, 63502,  2513,  3062, 60266,
        48289, 42818, 55050, 64289],
       [ 8996,  9735,   104, 62994, 62724, 62305, 64729, 64877, 57647,
        48808, 44028, 50366, 62740],
       [54548, 62064, 63499, 63532, 63263, 62868, 64223, 62898, 56373,
        48355, 44435, 45670, 59053],
       [44957, 55925, 62734, 62923, 64143, 63921, 64659, 61288, 55053,
        51041, 52352, 54525, 60475],
       [45572, 55193, 61504, 61722, 64101, 65035, 64374, 62440, 57375,
        57304, 64033, 64643, 65289],
       [46328, 55533, 62505, 62571, 63762, 64713, 63802, 62003, 59908,
        64619,  6630,  7772,  9474],
       [55115, 60252, 62899, 64386,   179, 64912, 59942, 53016, 50882,
        63002, 13442, 16664, 14581],
       [12003, 14106,  5484,   124,  1168, 65029, 55781, 47424, 44916,
        51874,  5423, 11687,  6979],
       [22113, 23323, 20820, 10620,  2143,   426, 54657, 47713, 45231,
        49208, 64607,  4038,  2332],
       [23825, 22371, 23116, 21133,  5977,  1507, 60034, 52637, 50492,
        55917,   318,  3126,  2865],
       [24628, 22861, 21292, 22229, 12367,  1582,   139, 61652, 60080,
          563,  2808,  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)
```

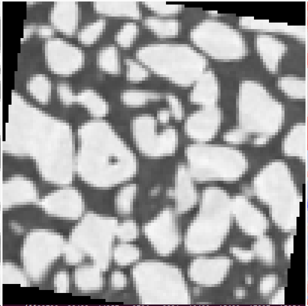
What about the evolution through time?

image 1



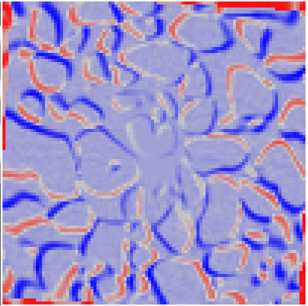
```
array([[11931, 11145, 11508, 12270, 12640, 12514, 13176, 14835, 20326,
        29225, 34191, 21655, 12285],
       [18759, 12223, 12034, 12864, 13021, 12939, 12926, 16789, 23822,
        29928, 33987, 26894, 13810],
       [27971, 15814, 12261, 12486, 13352, 12963, 12321, 17037, 25474,
        31316, 33720, 31330, 17174],
       [30987, 18643, 12490, 13615, 13259, 12608, 12316, 17200, 26466,
        31433, 32565, 32627, 24529],
       [30940, 20799, 13864, 13810, 12368, 12428, 12638, 14913, 22127,
        28696, 30402, 31378, 29082],
       [30308, 20819, 13442, 13171, 12174, 11788, 13367, 15867, 18360,
        20400, 23084, 24087, 21327],
       [28231, 16996, 12774, 11774, 10907, 11451, 17046, 24109, 26307,
        18931, 12581, 12947, 11612],
       [20042, 11917, 12057, 11458, 10142, 12744, 21917, 30117, 32083,
        26414, 13083,  9227,  9560],
       [11830, 10643, 11377, 11326,  9810, 12978, 24110, 30924, 32738,
        28492, 14357,  9786,  9831],
       [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, 11866],
       [13713, 11569, 12586, 13535, 11586, 10646, 10546, 10882, 10976,
        10456,  9832,  9996, 10933]], dtype=uint16)
```

image 2



```
array([[32738, 28492, 14357,  9786,  9831, 10480, 15689, 17897, 15056,
        11978, 11473, 11169, 11038],
       [27755, 21958, 12138, 10322, 10209,  9708, 12119, 16130, 15933,
        13200, 12479, 11724, 11014],
       [16983, 12342, 10224, 10482, 11079, 10295, 11008, 14399, 16311,
        14135, 12619, 11464, 10691],
       [10408,  9032,  9688, 11002, 11866, 10993, 11439, 12952, 15983,
        16938, 19381, 21616, 19468],
       [10976, 10456,  9832,  9996, 10933, 11927, 11476, 11817, 13966,
        20464, 28899, 30485, 28835],
       [11100, 10816, 10411, 10206, 10400, 10965, 11633, 12334, 12732,
        19483, 29714, 31859, 30801],
       [17810, 11712, 10137, 10624, 11086, 10827, 11452, 11589, 11653,
        16397, 26023, 29611, 26193],
       [32045, 26023, 17541, 11582, 11310, 12237, 12162, 12005, 11463,
        12752, 18506, 20914, 16539],
       [33943, 33966, 32197, 21946, 11953, 13404, 13231, 13101, 12433,
        12164, 13428, 13824, 12163],
       [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],
       [32681, 32889, 32693, 34115, 25019, 12476, 12798, 10718, 11111,
        12560, 12814, 12958, 11956],
       [26569, 32452, 33644, 31245, 17666, 11383, 12677, 11976, 11876,
        12189, 12612, 12771, 11369]], dtype=uint16)
```

residual



```
array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]], dtype=uint16)
```

The general idea of Digital Volume Correlation (DVC)

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

The general idea of Digital Volume Correlation (DVC)

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

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

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

The general idea of Digital Volume Correlation (DVC)

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

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

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

Deformation function in Spam (practical analysis)

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

where $\boldsymbol{\Phi}$ is **linear** and **homogeneous** accounting for:
translation, **rotation**, **normal** and **shear strain**

Lucas B. D., & Kanade T.: *An iterative image registration technique with an application to stereo vision*, IJCAI Proceedings (1981)

The general idea of Digital Volume Correlation (DVC) in Spam

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

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

The general idea of Digital Volume Correlation (DVC) in Spam

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

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

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

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

The general idea of Digital Volume Correlation (DVC) in Spam

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

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

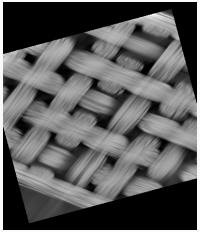
A Gauss-Newton iterative scheme is implemented based on a 1st order Taylor expansion of the updated deformed image $\widetilde{g}$ for **small** corrections of $\delta\boldsymbol{\Phi}$:

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

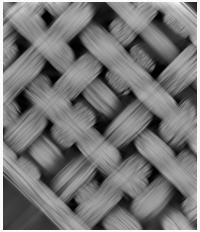
The system in a matrix-vector format:

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

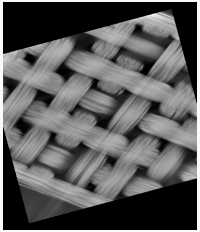
- $\boldsymbol{M}$ is the Hessian: gradient of g and
- $\boldsymbol{A}$ is the Jacobian: contains the difference between f and $\widetilde{g}$



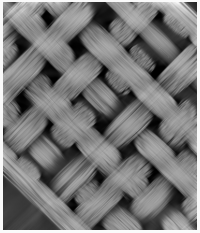
$g(\boldsymbol{x})$



$f(\boldsymbol{x})$

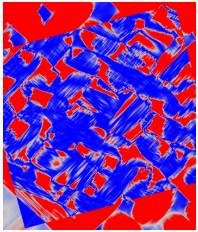


$g(\boldsymbol{x})$

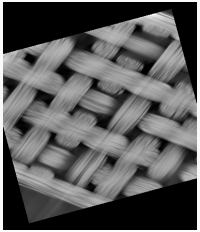


$f(\boldsymbol{x})$

$\|f(\boldsymbol{x}) - g(\boldsymbol{x})\|$

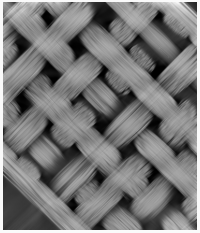


initial difference

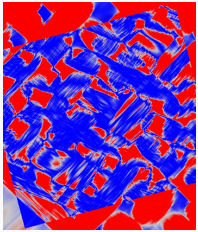


$g(\boldsymbol{x})$

1. Start with initial Φ

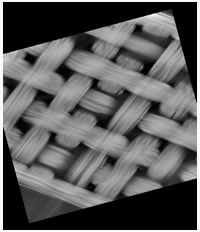


$f(\boldsymbol{x})$

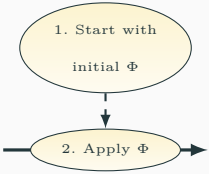


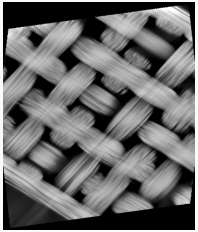
initial difference

Non-rigid registration with Spam

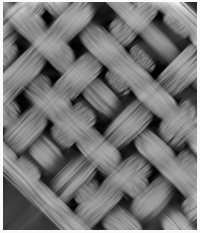


$g(\boldsymbol{x})$

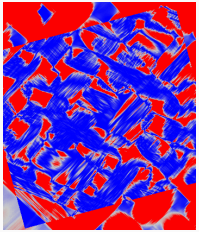




$g(\Phi \cdot \boldsymbol{x})$

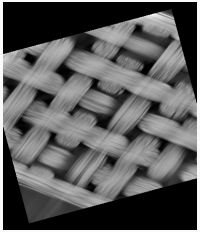


$f(\boldsymbol{x})$



initial difference

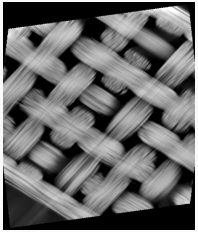
Non-rigid registration with Spam



$g(\boldsymbol{x})$

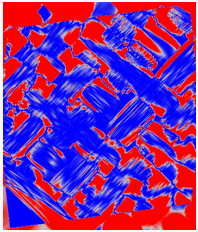
1. Start with initial Φ

2. Apply Φ

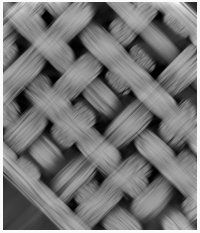


$g(\Phi \cdot \boldsymbol{x})$

3. Calculate $\|f(\boldsymbol{x}) - g(\Phi \cdot \boldsymbol{x})\|$

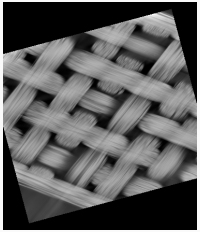


Difference



$f(\boldsymbol{x})$

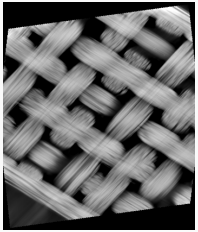
Non-rigid registration with Spam



$g(\boldsymbol{x})$

1. Start with initial Φ

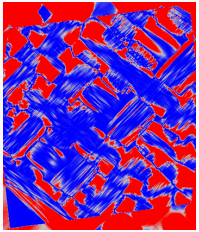
2. Apply Φ



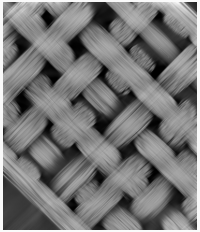
$g(\Phi \cdot \boldsymbol{x})$

3. Calculate $\|f(\boldsymbol{x}) - g(\Phi \cdot \boldsymbol{x})\|$

4. Correction $\delta\Phi$

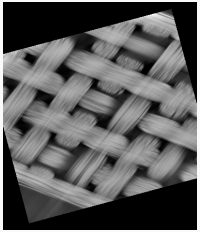


Difference



$f(\boldsymbol{x})$

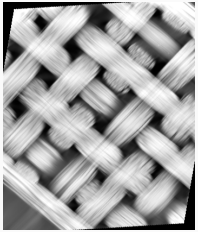
Non-rigid registration with Spam



$g(\boldsymbol{x})$

1. Start with initial Φ

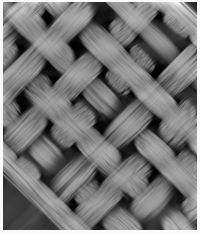
2. Apply Φ



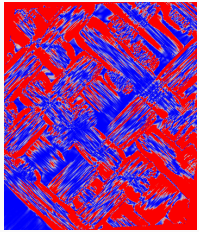
$g(\Phi \cdot \boldsymbol{x})$

3. Calculate $\|f(\boldsymbol{x}) - g(\Phi \cdot \boldsymbol{x})\|$

4. Correction $\delta\Phi$

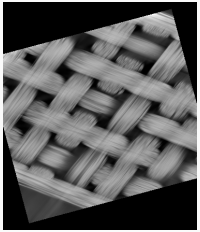


$f(\boldsymbol{x})$



Difference

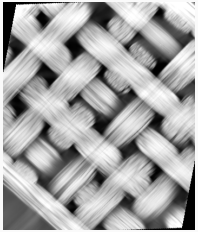
Non-rigid registration with Spam



$g(\boldsymbol{x})$

1. Start with initial Φ

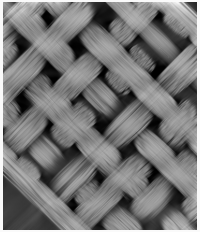
2. Apply Φ



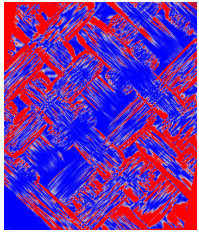
$g(\Phi \cdot \boldsymbol{x})$

3. Calculate $\|f(\boldsymbol{x}) - g(\Phi \cdot \boldsymbol{x})\|$

4. Correction $\delta\Phi$

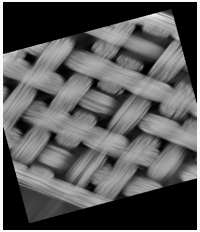


$f(\boldsymbol{x})$



Difference

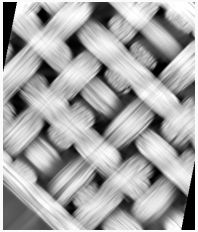
Non-rigid registration with Spam



$g(\boldsymbol{x})$

1. Start with initial Φ

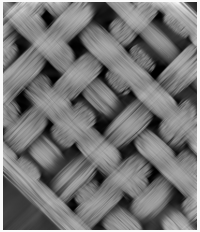
2. Apply Φ



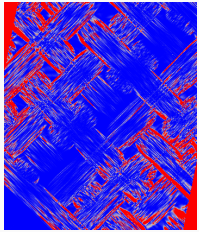
$g(\Phi \cdot \boldsymbol{x})$

3. Calculate $\|f(\boldsymbol{x}) - g(\Phi \cdot \boldsymbol{x})\|$

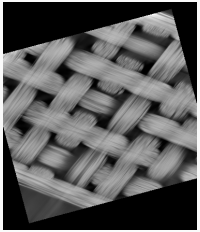
4. Correction $\delta\Phi$



$f(\boldsymbol{x})$



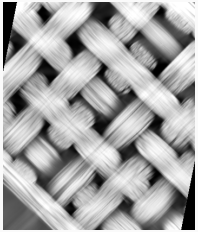
Difference



$g(\boldsymbol{x})$

1. Start with initial Φ

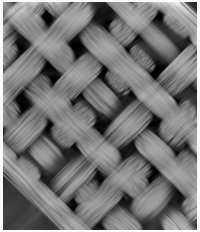
2. Apply Φ



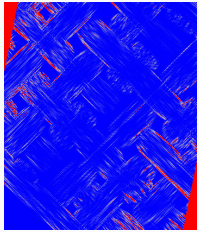
$g(\Phi \cdot \boldsymbol{x})$

3. Calculate $\|f(\boldsymbol{x}) - g(\Phi \cdot \boldsymbol{x})\|$

4. Correction $\delta\Phi$

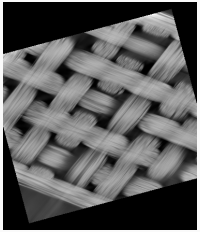


$f(\boldsymbol{x})$



Difference

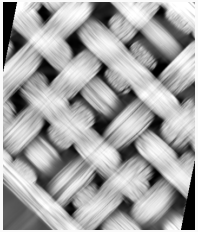
Non-rigid registration with Spam



$g(\boldsymbol{x})$

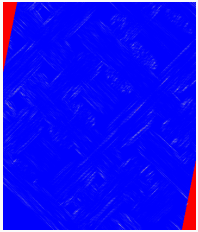
1. Start with initial Φ

2. Apply Φ



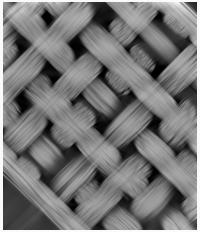
$g(\Phi \cdot \boldsymbol{x})$

3. Calculate $\|f(\boldsymbol{x}) - g(\Phi \cdot \boldsymbol{x})\|$



Difference

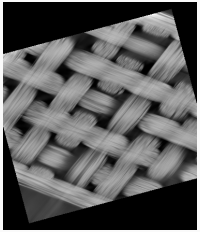
4. Correction $\delta\Phi$



$f(\boldsymbol{x})$

$$\Phi = \begin{pmatrix} 1. & 0. & 0. & 0. \\ 0. & 0.724 & -0.259 & 0. \\ 0. & 0.194 & 0.966 & 0. \\ 0. & 0. & 0. & 1. \end{pmatrix}$$

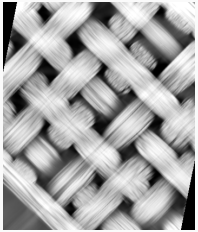
Non-rigid registration with Spam



$g(\boldsymbol{x})$

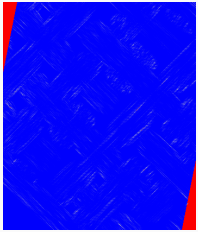
1. Start with initial Φ

2. Apply Φ



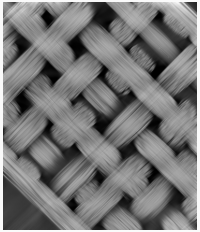
$g(\Phi \cdot \boldsymbol{x})$

3. Calculate $\|f(\boldsymbol{x}) - g(\Phi \cdot \boldsymbol{x})\|$



Difference

4. Correction $\delta\Phi$



$f(\boldsymbol{x})$

$$\Phi = \begin{pmatrix} 1. & 0. & 0. & 0. \\ 0. & 0.724 & -0.259 & 0. \\ 0. & 0.194 & 0.966 & 0. \\ 0. & 0. & 0. & 1. \end{pmatrix}$$

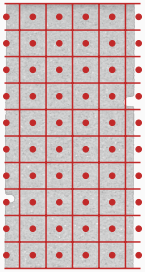
$$\mathcal{T}(\widetilde{\Phi}) = \frac{1}{2} \sum_{\boldsymbol{x} \in \Omega} \left(f(\boldsymbol{x}) - g(\widetilde{\Phi} \cdot \boldsymbol{x}) \right)^2$$

How to get a deformation field?

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

How to get a deformation field?

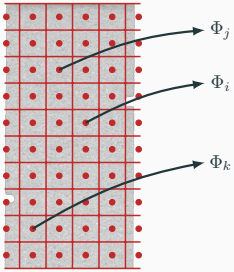
$$\mathcal{T}(\widetilde{\boldsymbol{\Phi}}) = \frac{1}{2} \sum_{\boldsymbol{x} \in \Omega} \left(f(\boldsymbol{x}) - g(\widetilde{\boldsymbol{\Phi}} \cdot \boldsymbol{x}) \right)^2$$



Correlation windows
defined on a regular grid

How to get a deformation field?

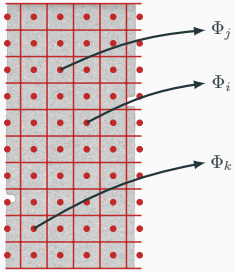
$$\mathcal{T}(\widetilde{\boldsymbol{\Phi}}) = \frac{1}{2} \sum_{\boldsymbol{x} \in \Omega} \Big(f(\boldsymbol{x}) - g(\widetilde{\boldsymbol{\Phi}} \cdot \boldsymbol{x}) \Big)^2$$



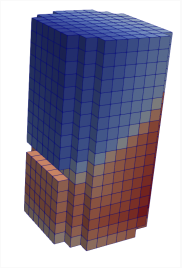
Correlation windows
defined on a regular grid

How to get a deformation field?

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



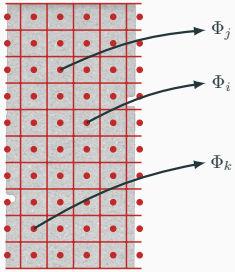
Correlation windows
defined on a regular grid



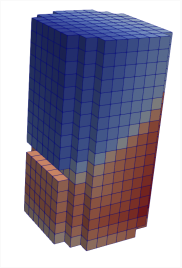
Displacement field

How to get a deformation field?

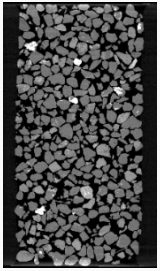
$$\mathcal{T}(\widetilde{\boldsymbol{\Phi}}) = \frac{1}{2} \sum_{\boldsymbol{x} \in \Omega} \left(f(\boldsymbol{x}) - g(\widetilde{\boldsymbol{\Phi}} \cdot \boldsymbol{x}) \right)^2$$



Correlation windows
defined on a regular grid

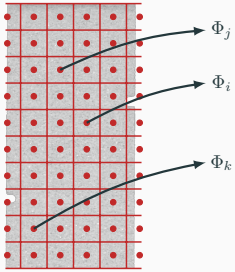


Displacement field

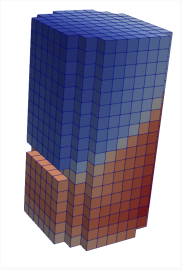


How to get a deformation field?

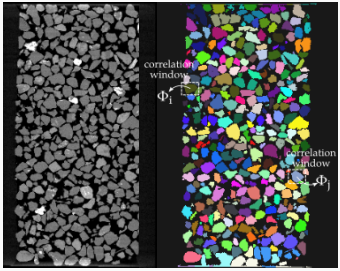
$$\mathcal{T}(\widetilde{\mathbf{\Phi}}) = \frac{1}{2} \sum_{\boldsymbol{x} \in \Omega} \left(f(\boldsymbol{x}) - g(\widetilde{\mathbf{\Phi}} \cdot \boldsymbol{x}) \right)^2$$



Correlation windows
defined on a regular grid



Displacement field



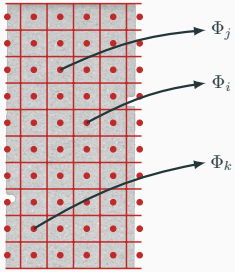
Correlation windows
defined on labels

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)

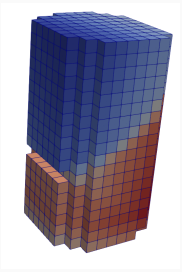
Andò E. *et al.*: *Experimental micromechanics: grain-scale observation of sand deformation*, Géotechnique Letters (2012)

How to get a deformation field?

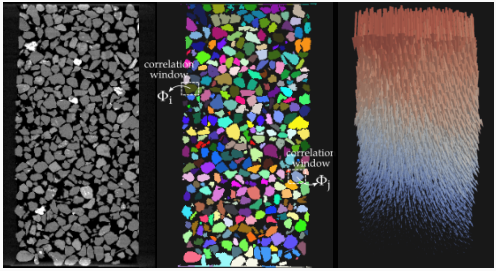
$$\mathcal{T}(\widetilde{\boldsymbol{\Phi}}) = \frac{1}{2} \sum_{\boldsymbol{x} \in \Omega} \left(f(\boldsymbol{x}) - g(\widetilde{\boldsymbol{\Phi}} \cdot \boldsymbol{x}) \right)^2$$



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)

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

How to produce these deformation fields with spam

1. **Initial registration** `script` called from `bash`

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

2. **Regular grid DVC** `script` called from `bash`

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

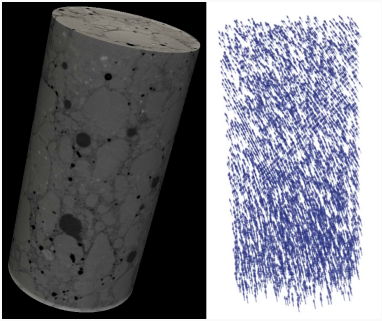
3. **Strains** `script` called from `bash`

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

Read the Docs https://www.spam-project.dev/docs/scripts.html#image-correlation-script-flowchart

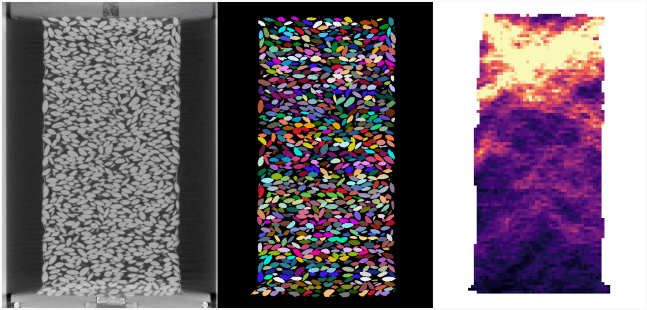
How to get a deformation field through time?

DVC on a regular grid



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

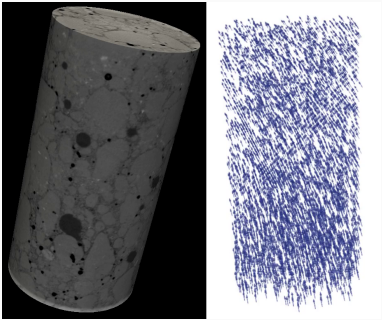
DVC on labelled objects



[Pinzón G. (2023) Gran. Matter]

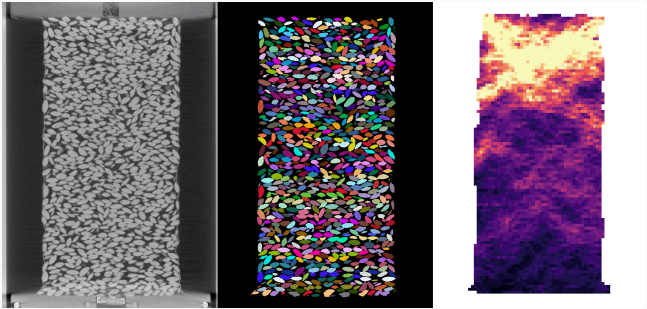
How to get a deformation field through time?

DVC on a regular grid



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

DVC on labelled objects

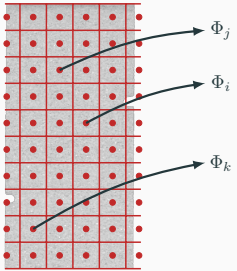


[Pinzón G. (2023) Gran. Matter]

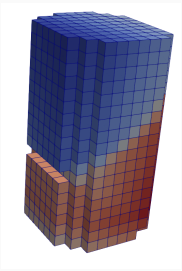
Both are considered as local correlation approaches

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.



Regular grid

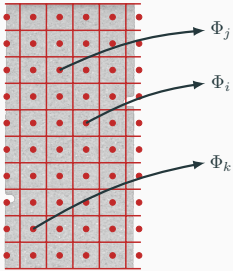


Displacement field

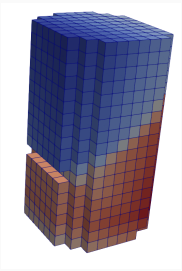
$$\mathcal{T}(\widetilde{\boldsymbol{\Phi}}) = \frac{1}{2} \sum_{\boldsymbol{x} \in \Omega} \left(f(\boldsymbol{x}) - g(\widetilde{\boldsymbol{\Phi}} \cdot \boldsymbol{x}) \right)^2$$

Global vs Local DVC approaches

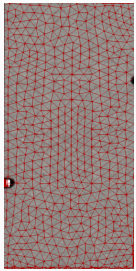
The paradigm behind the global correlation is to transform an image with a **displacement field** which support is a Finite Element mesh instead of a set of linear operators Φ.



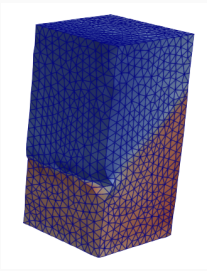
Regular grid



Displacement field



FE mesh



Displacement field

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

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

Hild F. and Roux S.: *Comparison of local and global approaches to digital image correlation*, Experimental Mechanics (2012)

A Finite Element mesh to transform an image

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

$$\Phi_{\mathrm{c}} = \sum_{\Omega} \|g(\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x})) - f(\boldsymbol{x})\|_2$$

where:

- $\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{N}(\boldsymbol{x})\boldsymbol{d}$ 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:

$$\Phi_{\mathrm{c}} = \sum_{\Omega} \|g(\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x})) - f(\boldsymbol{x})\|_2$$

In it's weak form can be solved with a Newton-Raphson method:

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

where:

- $\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{N}(\boldsymbol{x})\boldsymbol{d}$ 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:

$$\Phi_{\mathrm{c}} = \sum_{\Omega} \|g(\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x})) - f(\boldsymbol{x})\|_2$$

In it's weak form can be solved with a Newton-Raphson method:

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

where:

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

Mechanical regularization

Mechanically driven Tikhonov regularization

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

$$\begin{cases} \Phi_c(\boldsymbol{d}) = \sum_\Omega \|g(\boldsymbol{x}+\boldsymbol{u}(\boldsymbol{x})) - f(\boldsymbol{x})\|_2 \end{cases}$$

The regularised problem becomes:

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

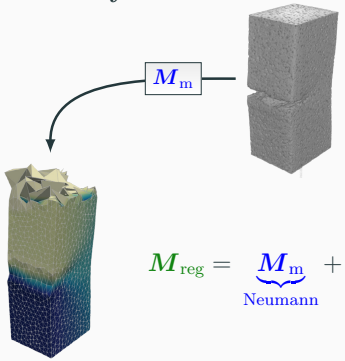
Mendoza A., Neggers J., Hild F., Roux S.: *Complete mechanical regularization applied to digital image and volume correlation*, Computer Methods in Applied Mechanics and Engineering (2019)

Mechanical regularization

Mechanically driven Tikhonov regularization

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

$$\begin{cases} \Phi_{\mathrm{c}}(\boldsymbol{d}) = \sum_{\Omega} \|g(\boldsymbol{x}+\boldsymbol{u}(\boldsymbol{x})) - f(\boldsymbol{x})\|_2 \\ \Phi_{\mathrm{m}}(\boldsymbol{d}) = \|\boldsymbol{K}\boldsymbol{u} - \boldsymbol{f}\|_2 \quad \text{(Neumann)} \end{cases}$$



The regularised problem becomes:

$$(\boldsymbol{M}_c + \boldsymbol{M}_{\mathrm{reg}})\,\delta\boldsymbol{d} = \boldsymbol{b} - \boldsymbol{M}_{\mathrm{reg}}\boldsymbol{d}$$

$$\boldsymbol{M}_{\mathrm{reg}} = \underbrace{\boldsymbol{M}_{\mathrm{m}}}_{\text{Neumann}} + \cdots$$

Mendoza A., Neggers J., Hild F., Roux S.: *Complete mechanical regularization applied to digital image and volume correlation*, Computer Methods in Applied Mechanics and Engineering (2019)

Mechanical regularization

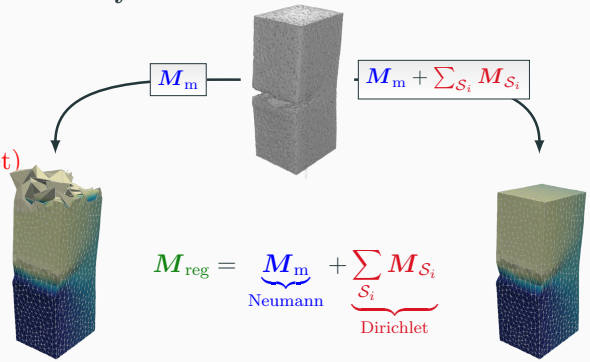
Mechanically driven Tikhonov regularization

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

$$\begin{cases} \Phi_{\mathrm{c}}(\boldsymbol{d}) = \sum_{\Omega} \|g(\boldsymbol{x}+\boldsymbol{u}(\boldsymbol{x})) - f(\boldsymbol{x})\|_2 \\ \Phi_{\mathrm{m}}(\boldsymbol{d}) = \|\boldsymbol{K}\boldsymbol{u} - \boldsymbol{f}\|_2 \quad \text{(Neumann)} \\ \Phi_{\mathcal{S}_i}(\boldsymbol{d}) = \int_{\mathcal{S}_i} \|\boldsymbol{\nabla}(\boldsymbol{\sigma}\cdot\boldsymbol{n})\|_2 d\boldsymbol{x} \quad \text{(Dirichlet)} \end{cases}$$

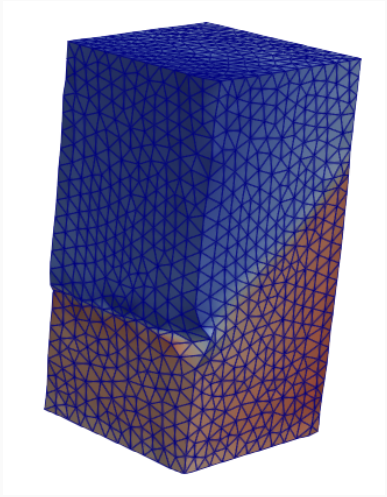
The regularised problem becomes:

$$(\boldsymbol{M}_c + \boldsymbol{M}_{\mathrm{reg}})\,\delta\boldsymbol{d} = \boldsymbol{b} - \boldsymbol{M}_{\mathrm{reg}}\boldsymbol{d}$$



Mendoza A., Neggers J., Hild F., Roux S.: *Complete mechanical regularization applied to digital image and volume correlation*, Computer Methods in Applied Mechanics and Engineering (2019)

Regularized Global DVC



Advantages

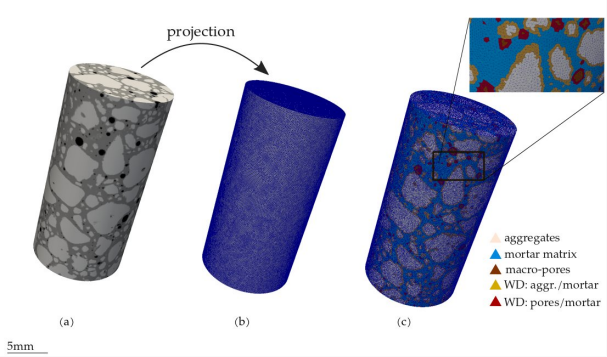
- Continuity of sought displacement field
- Natural regularization
- Mechanically admissible solution
- Straightforward link with simulations
- Can work for non-elastic problems

Global DVC in Spam

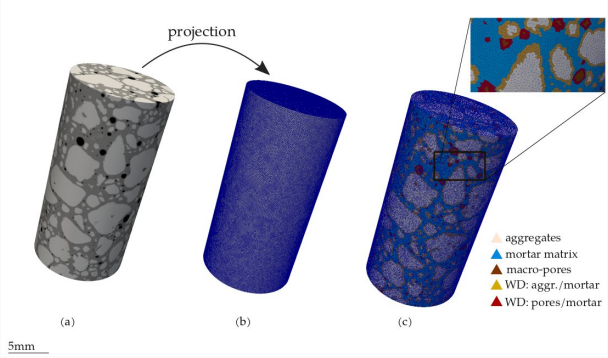
- Recently implemented
- Looking for nice data to test it!

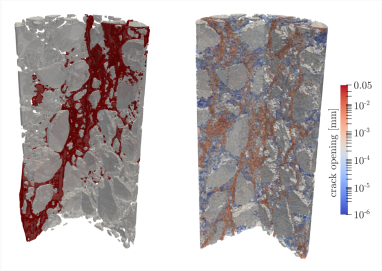
https://www.spam-project.dev/docs/spam.DIC.html#module-spam.DIC.globalDVC

Bridging the gap between images and FE models



Bridging the gap between images and FE models

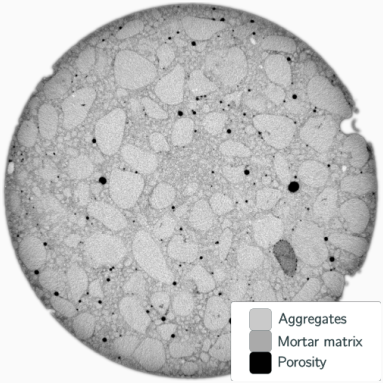




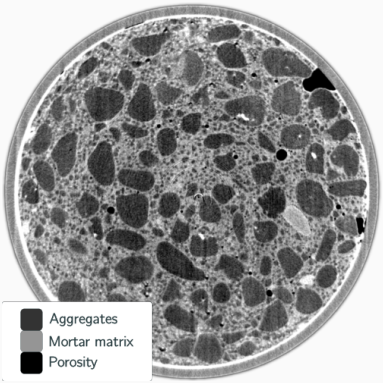
Stamati O., Roubin E., Andò E., Malecot Y.: *Tensile failure of micro-concrete: from mechanical tests to fe meso-model with the help of x-ray tomography*, Meccanica (2019)

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

Multiple modalities

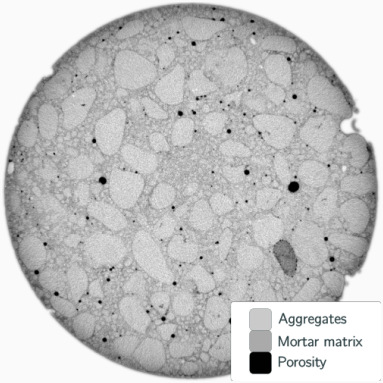


X-ray scan

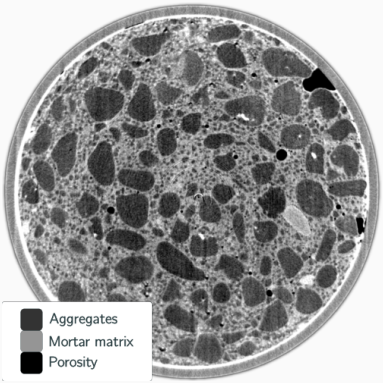


Neutrons scan from D50 (NeXT, ILL)

Multiple modalities



X-ray scan



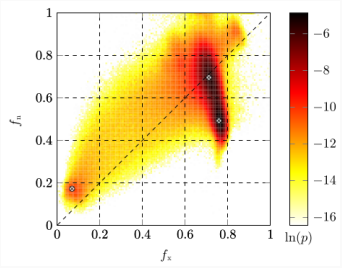
Neutrons scan from D50 (NeXT, ILL)

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

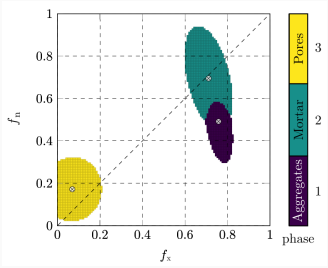
Multimodal registration

A solution

Construct a functional $\mathbf{\Phi}(f_\mathrm{x}, f_\mathrm{n})$ based the joint histogram $p(f_\mathrm{x}, f_\mathrm{n})$.



Joint histogram $p(f_\mathrm{x}, f_\mathrm{n})$

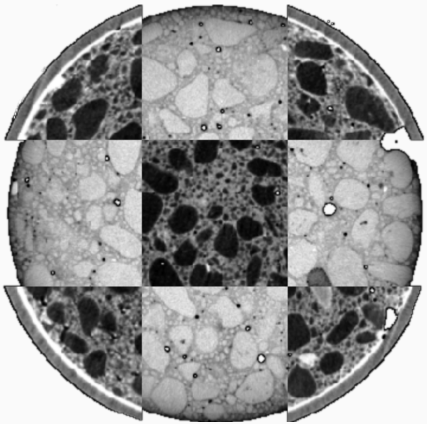


Phase repartition

Tudisco, E., Jailin 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)

Multimodal registration



MMR in Spam

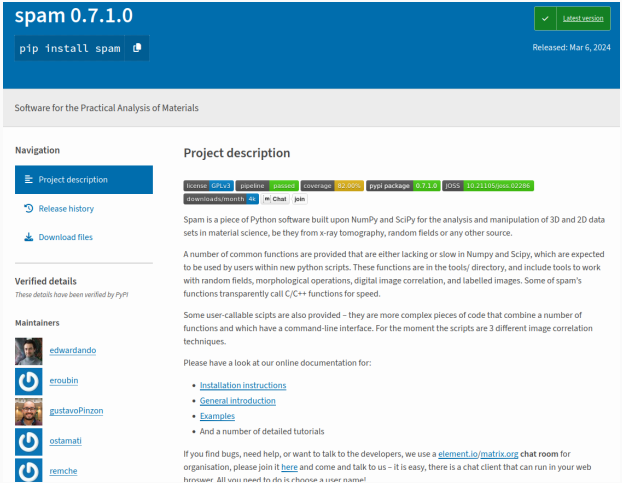
- Online tutorials and examples
- Used in many published works

Going further

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

General information

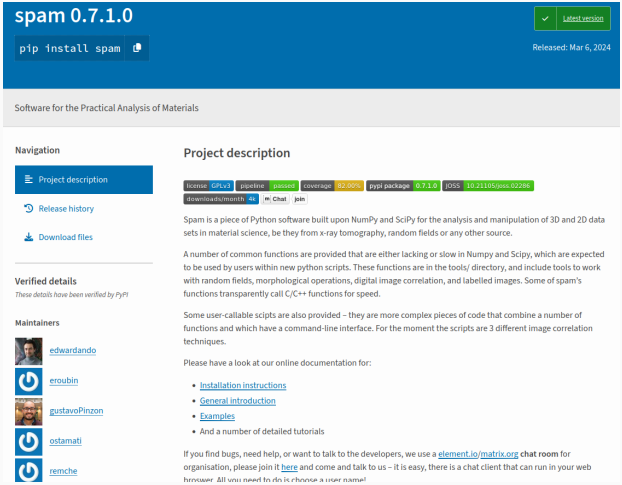
python package

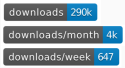


Stamati, Andò, Roubin: *Journal of Open Source Software*, 2020 (Citations: 110)

https://pepy.tech/project/spam

python package





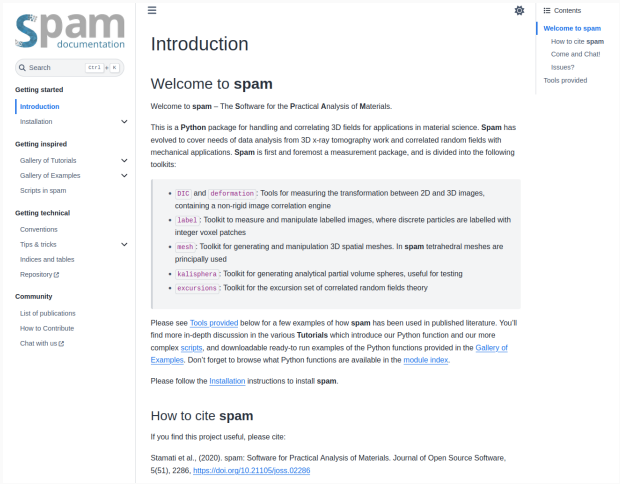
Stamati, Andò, Roubin: *Journal of Open Source Software*, 2020 (Citations: 110)

Easy to use!!

```
pip install spam
```

https://pepy.tech/project/spam

documentation

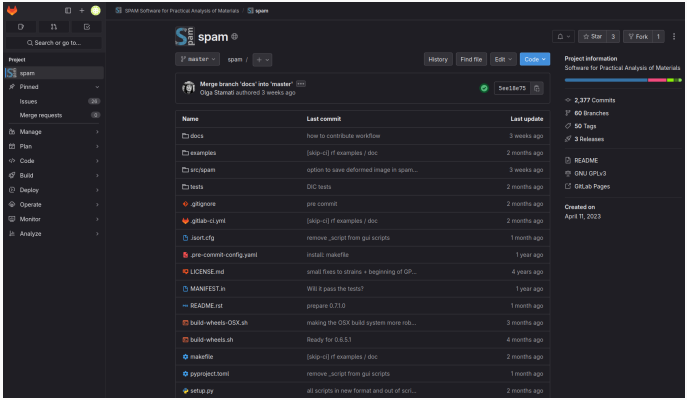


- Installation instructions
- Tutorials
- Examples
- Module indices
- Code repository
- Communication
- How to contribute

Read the Docs

https://www.spam-project.dev/docs/intro.html

repository

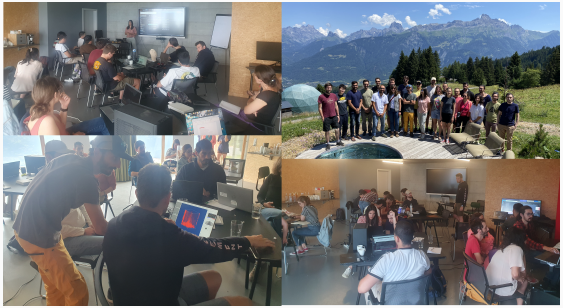


- Source code
- 93% test coverage
- Reporting issues
- Contributing!

`https://gitlab.com/spam-project/spam`

workshops

Valais, Switzerland 18-21/07/2023



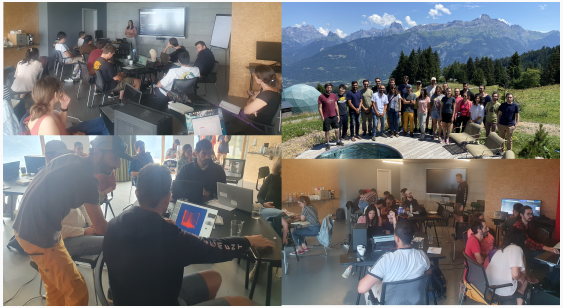
https://www.spam-project.dev/workshops/2023/07/

Villar d'Arène, France 6-9/10/2024

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

workshops

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