Projet Collaboratif FUI

GigaQuant

Nouvelles méthodologies pour le traitement d'images 3D

Journée Tomo3D Precend – 14 mai 2019

Reactiv'IP
163 cours Berriat
38000 Grenoble - France
Web: www.reactivip.com

Contact:
M. Laurent Bernard
Laurent.bernard@reactivip.com
Tél.: +33 (0)4 58 00 38 85

Agenda

La société Reactiv'IP

Reactiv'IP

Société dédiée au traitement d'images

Développement d'applications

Réalisation d'applications sur-mesure à partir de spécifications ou d'une publication scientifique.

Logiciels

Editeur des logiciels IPSDK, Astra MetalloBox, FluoBox

Formations

Formations aux logiciels Astra, Image-Pro, IPSDK, DragonFly et Visilog.

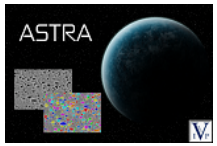
Expertises

Société à taille humaine à l'écoute de vos problématiques pour définir ensemble les solutions de traitement d'images les plus adaptées.

Basée à Grenoble

Reactiv'IP - Logiciels

- **Librairie IPSDK**
  - Kit de développement optimisé dédié au traitement d'images 2D et 3D



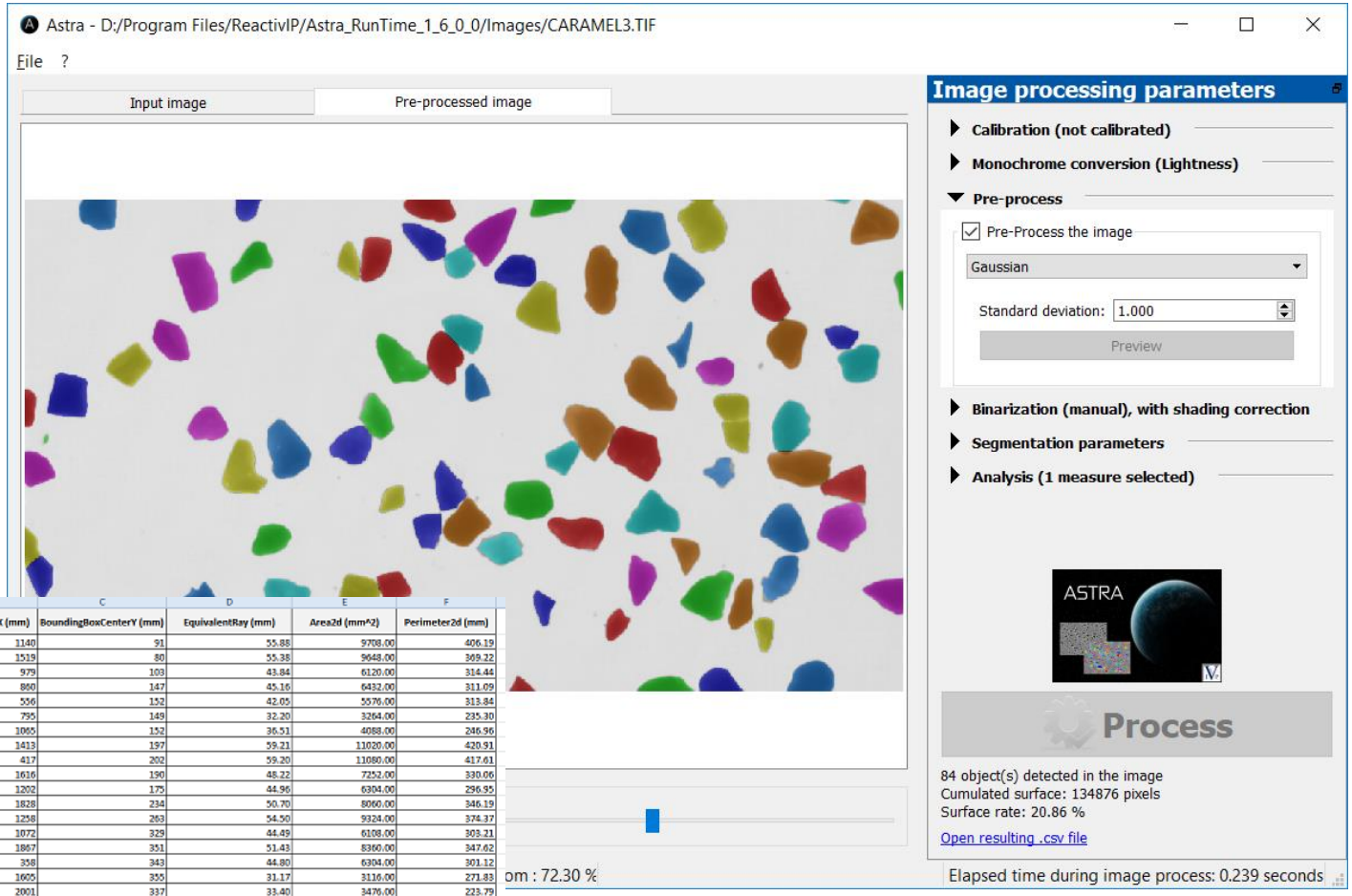
- **Logiciel ASTRA**
  - Pour la détection et la mesure automatique d'objets

- **Application MetalloBox (OpenSource)**
  - Pour la mesure de taille de grains et taux de phase en métallurgie

- **FluoBox (OpenSource)**
  - Logiciel dédié à la quantification de marqueurs

ASTRA

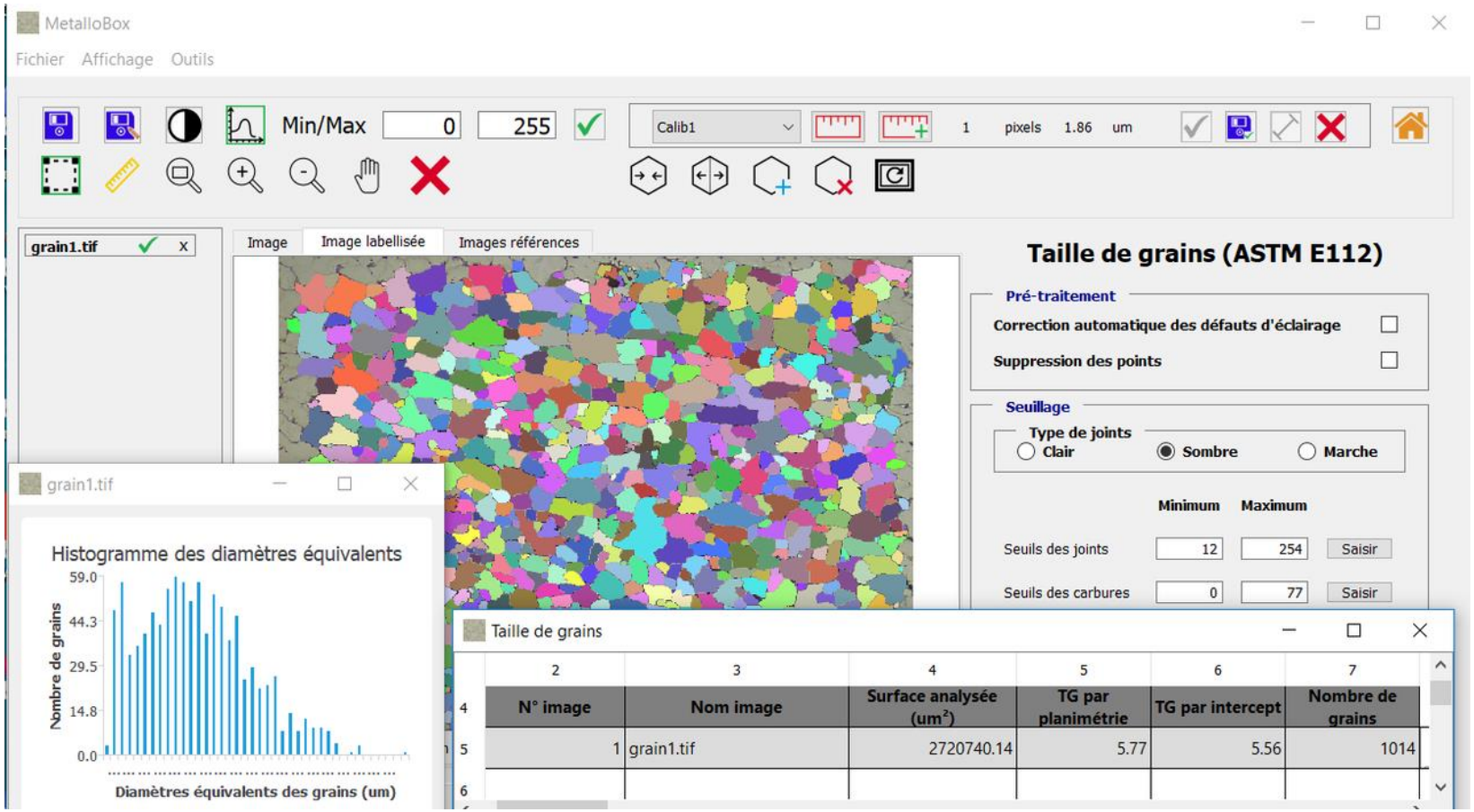
- **Logiciel ASTRA (C++)**
  Comptage et mesure automatique d'objets 2D



object identifier	BoundingBoxCenterX (mm)	BoundingBoxCenterY (mm)	EquivalentRay (mm)	Area2d (mm^2)	Perimeter2d (mm)
1	1140	91	55.88	9708.00	406.19
2	1519	80	55.38	9648.00	369.22
3	979	103	43.84	6120.00	314.44
4	860	147	45.16	6432.00	311.09
5	556	152	42.05	5576.00	313.84
6	795	149	32.20	3264.00	235.30
7	1065	152	36.51	4088.00	246.96
8	1413	197	59.21	11020.00	420.91
9	417	202	59.20	11080.00	417.61
10	1616	190	48.22	7252.00	330.06
11	1202	175	44.96	6304.00	296.95
12	1828	234	50.70	8060.00	346.19
13	1258	263	54.50	9324.00	374.37
14	1072	329	44.49	6108.00	303.21
15	1867	351	51.43	8360.00	347.62
16	358	343	44.80	6304.00	301.12
17	1605	355	31.17	3116.00	271.83
18	2001	337	33.40	3476.00	223.79
19	1031	387	50.14	7924.00	346.09
20	927	388	52.13	8624.00	389.95

MetalloBox

- **Logiciel MetalloBox (Python, Open-source)**
  Mesures pour la métallographie (Taille de grains, taux de phases)



Reactiv'IP - Services

- **Développements spécifiques**
  - Réalisation d'application sur cahier des charges
  - Réalisation de traitements complémentaires
  - Optimisation d'algorithmes (outils internes clients ou open source)
- **Support technique**
- **Formations**
  - Formation au traitement d'images
  - Formation aux différents logiciels (IPSDK, Visilog, Image-Pro, ASTRA)
- **Participation à des projets de recherche**
- **Dépouillement de données images**

Exemples

Outils puissants de filtrage d'images

IPSDK possède de nombreuses fonctionnalités de filtrage et de débruitage. Parmi elles, on trouve notamment une évolution du célèbre **Non Local Means** et la diffusion anisotrope. La figure suivante montre le résultat obtenu sur une image très bruitée acquise en microscopie électronique.

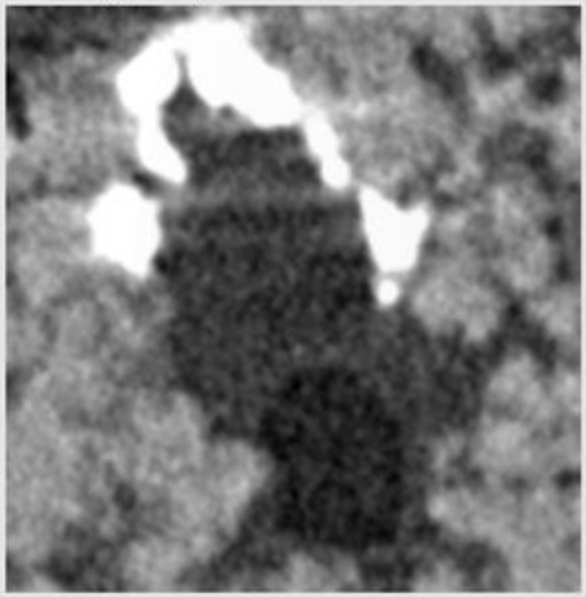


Image originale

Diffusion anisotrope

Exemples

Caractérisation de porosités sur image de FIB (Focused Ion Beam)

IPSDK possède des fonctionnalités de caractérisation de matériaux. Il est ainsi possible de déterminer les porosités d'un matériau en utilisant les fonctionnalités de débruitage, de binarisation, de séparation et de labellisation. Il est également possible d'effectuer des mesures sur les éléments identifiés.

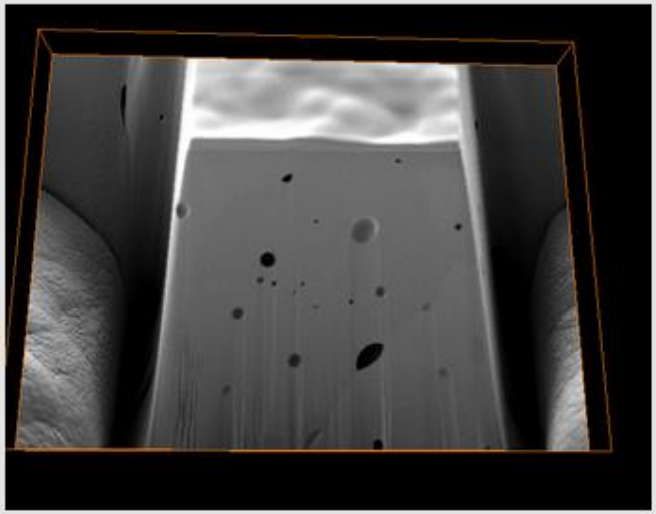
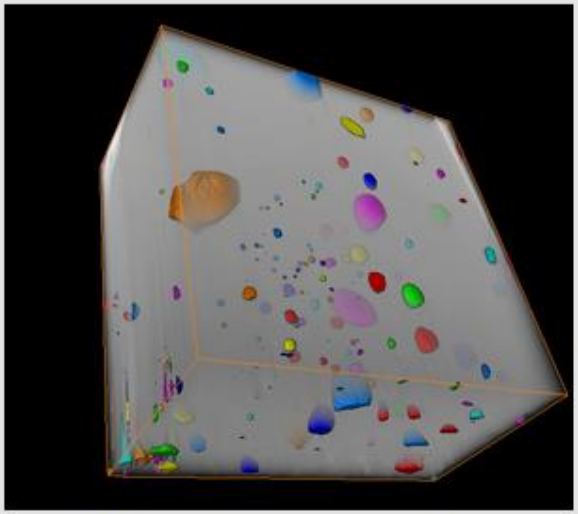
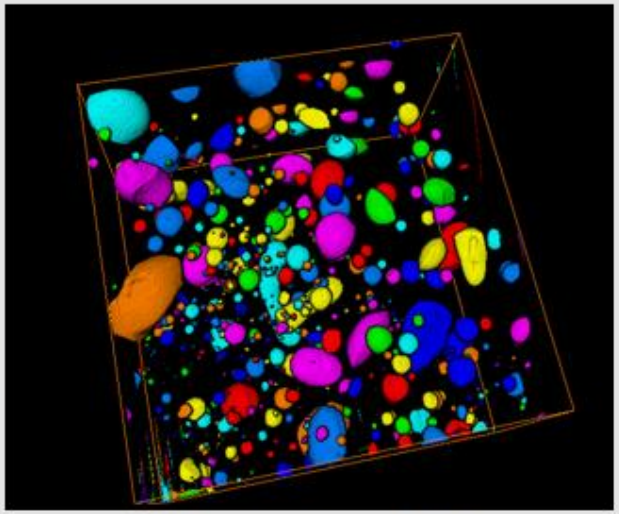


Image originale



Volume avec surimpression des trous

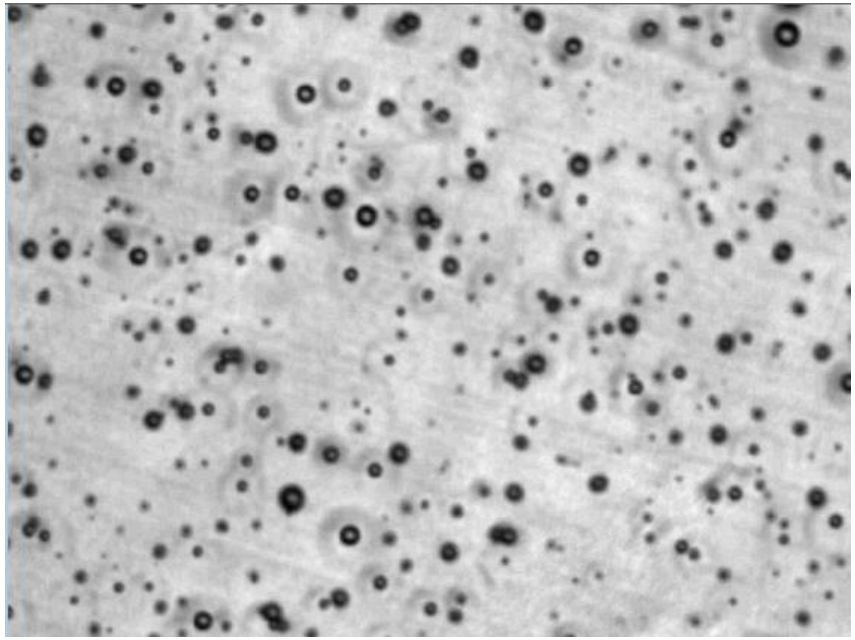


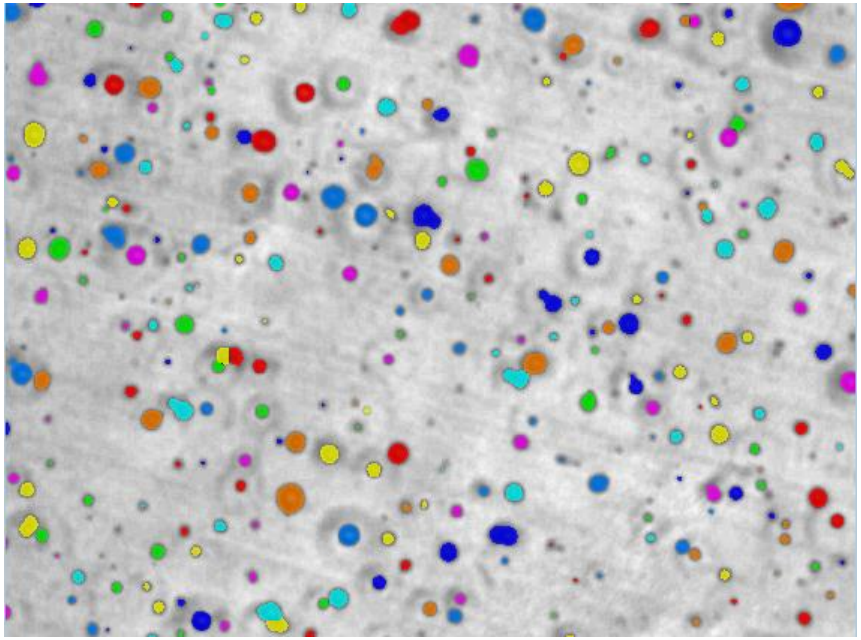
Affichage des trous seulement

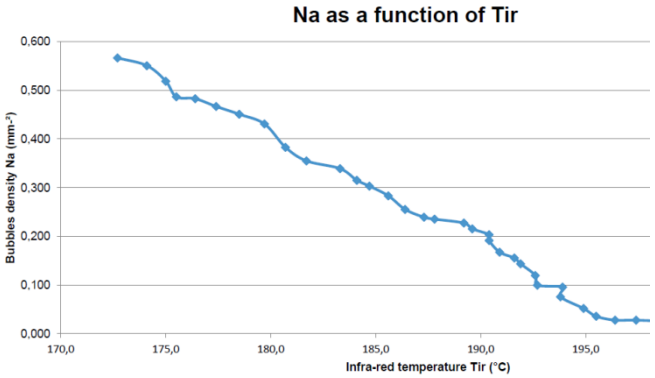
- Fonctionnalités de recalage plan à plan basé sur les descripteurs
- Correction de dérive (shrinkage)

Exemples

Rotomoulage: Evolution de la présence de bulles suivant T°



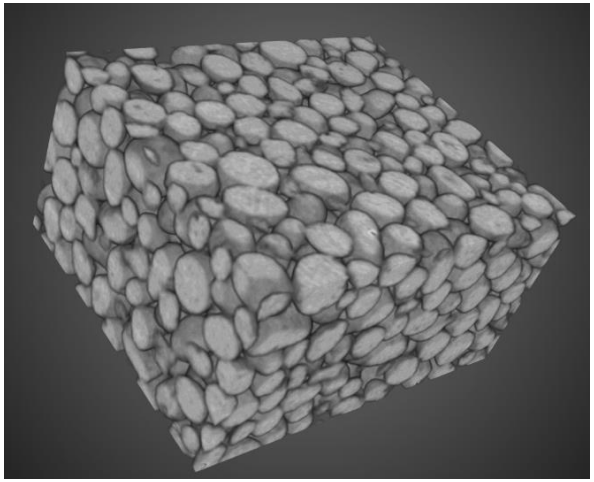


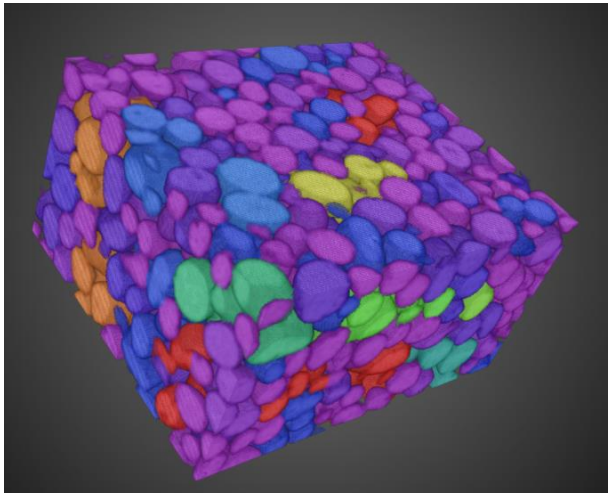


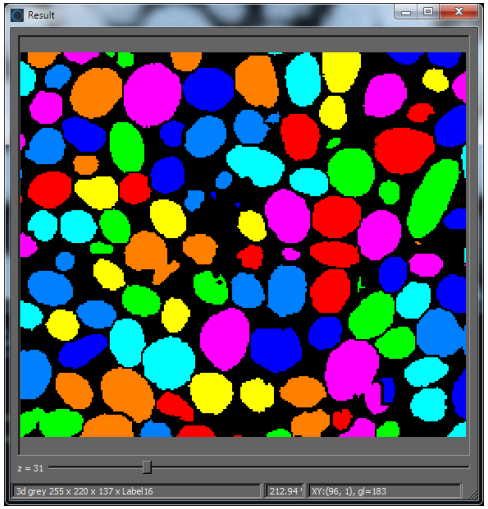
TOTAL
COMMITTED TO BETTER ENERGY

Utilisation avec DragonFly

Segmentation de roches (Tomo)
Utilisation d'IPSDK en Python dans ORS DragonFly

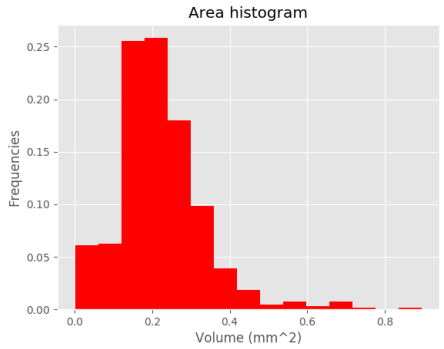







```
In [6]: run "ExempleSeparateCsv.py"
```

	A	B	C	D
1	Area3dMsr	EquivalentRayMsr	MaxFeretDiameterMsr	MeanMsr
2	mm^2	mm	mm	
3	0.103179246	0.068424605	0.292135455	46233.10469
4	0.236791934	0.11872479	0.403710365	49070.58513
5	0.000173205	0.003413919	0.009948833	33320
6	0.001968142	0.01072442	0.044564366	35548.125
7	0.257163671	0.131688702	0.336424232	49945.86158
8	0.126663054	0.088504885	0.272377253	49025.93707
9	0.239933326	0.124276322	0.394010693	48990.05487
10	0.277188757	0.130159791	0.401493192	49712.53877
11	0.188664102	0.103195935	0.324559927	48782.81075
12	0.132468191	0.091886541	0.290334463	51364.05809
13	0.020543085	0.035384779	0.120274412	47026.50515
14	0.221403437	0.120429843	0.288072217	50052.51757
15	0.161873591	0.099156354	0.301295735	48785.31825
16	0.225750093	0.122107957	0.317851409	48823.76682
17	0.114474649	0.079556214	0.245292783	47544.45186



IPSDK
Smart Image Processing

Reactiv'IP
SMART IMAGE PROCESSING

Projet GigaQuant

bpifrance

La Région
Auvergne-Rhône-Alpes

Projet FUI GigaQuant

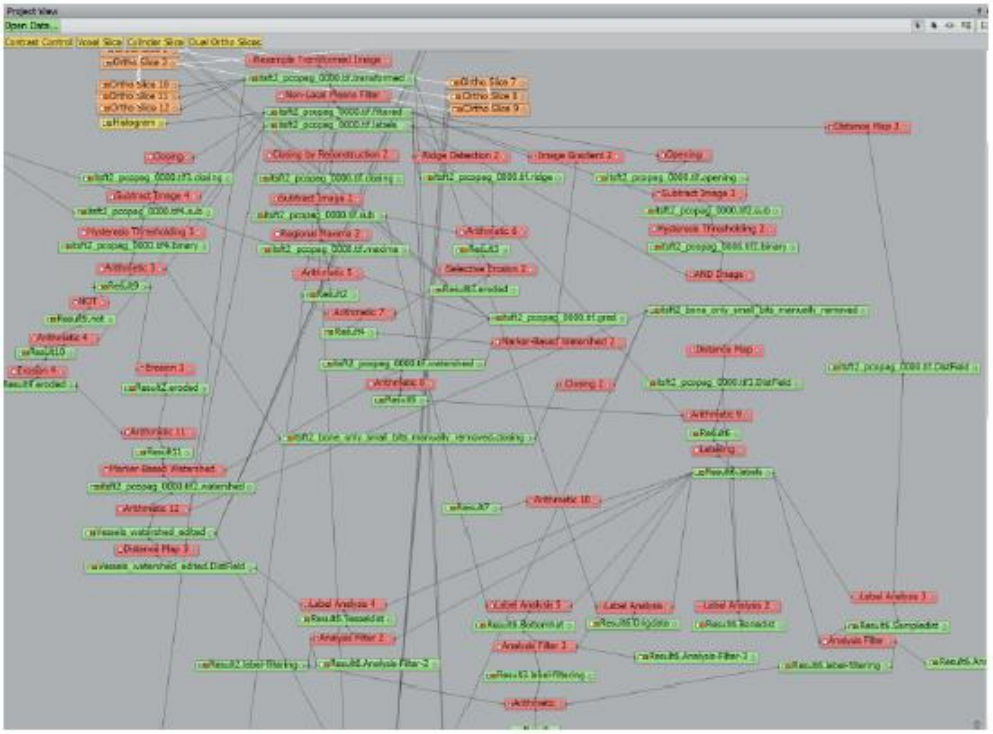
Développement d'outils logiciels pour la manipulation et le traitements d'images 3D de grandes tailles (sup à 1 Go)

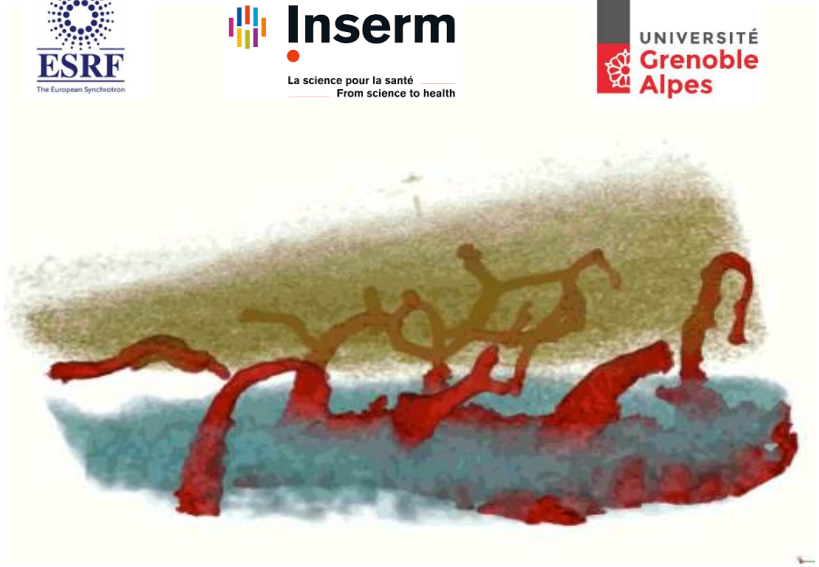
Partenaires:

- **Reativ'IP**: Pilote – Développement d'IPSDK
- **Digisens**: Développement de Digi-XCT (Visualisation & Reconstruction)
- **ESRF/UGA**: Développement de nouveaux algorithmes de reconstruction et d'analyse
- **AMU**: Développement du logiciel eMorph pour l'analyse des milieux poreux.
- **Groupe Total**: End user

Objectif: Accélérer tous ces produits à l'aide de la librairie IPSDK Mise à disposition des algorithmes nécessaires pour ces problématiques

Projet FUI GigaQuant





```
# opening of input image
greyImg = PyIPSDK.loadTiffImageFile( inputImgPath, PyIPSDK.eTiffDirectoryMode.eTDM_Volume)

 # definition of used structuring element
inSE = PyIPSDK.circularSEXYInfo(11)

# ouverture en XY pour renforcer le délaminage
openImg = morpho.opening2dImg(greyImg, inSE)

# fermeture lineaire en Z pour fermer le délaminage
inLE = PyIPSDK.linearSEXYZInfo(0,0,10)
closingImg = morpho.closing3dImg(openImg, inLE)

subImg = arithm.subtractImgImg(closingImg, openImg)

# seuillage du delaminage
binImg = bin.thresholdImg(subImg, 3300, 65535)

# nettoyage
inSE = PyIPSDK.sphericalSEXYZInfo(2)
erodeImg = morpho.erode3dImg(binImg, inSE)
binImgClean = advmorpho.binaryReconstruction3dImg(binImg, erodeImg)
biggest = advmorpho.keepBigShape3dImg(binImgClean, 1)

outImg = gblmsr.seqProjectionImg(ImgSeq, PyIPSDK.eProjStatType.ePST_Sum)
gui.displayImg(outImg, 'outImg' )

PyIPSDK.saveTiffImageFile(outputImgPath5, outImg)
```

Simplification du process

&

Accélération du temps de traitement

(de plusieurs heures à quelques minutes)

Projet FUI GigaQuant

Simplification du process & **Accélération du temps de traitement (de plusieurs heures à quelques minutes)**

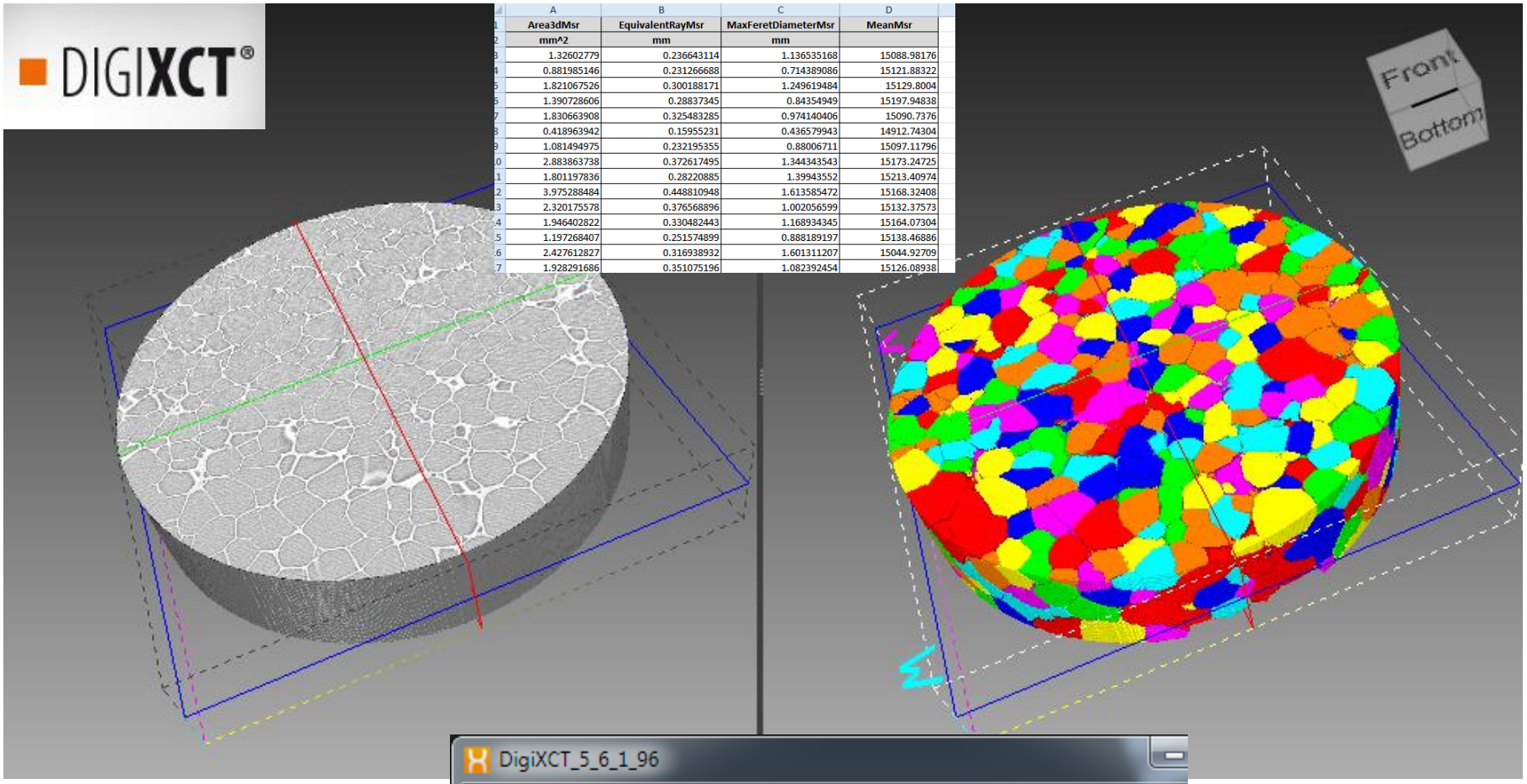
Objectifs:

- Diviser au minimum par dix les temps de traitement
- Pouvoir lancer des traitements en batch sur des machines dédiées
- Autoriser les traitements aussi sur GPU
- Rendre le logiciel eMorph utilisable sur de gros volumes de données
- Autoriser le paramétrage fin de l'ensemble des algorithmes (pas de boites noires)
- Rendre IPSDK intégrable facilement dans d'autres produits (Voxaya, RX Solutions, …)

Utilisation avec Digi-XCT de Digisens

DIGIXCT®

	A	B	C	D
1	Area3dMsr	EquivalentRayMsr	MaxFeretDiameterMsr	MeanMsr
2	mm^2	mm	mm	
3	1.32602779	0.236643114	1.136535168	15088.98176
4	0.881985146	0.231266688	0.714389086	15121.88322
5	1.821067526	0.300188171	1.249619484	15129.8004
6	1.390728606	0.28837345	0.84354949	15197.94838
7	1.830663908	0.325483285	0.974140406	15090.7376
8	0.418963942	0.15955231	0.436579943	14912.74304
9	1.081494975	0.232195355	0.88006711	15097.11796
10	2.883863738	0.372617495	1.344343543	15173.24725
11	1.801197836	0.28220885	1.39943552	15213.40974
12	3.975288484	0.448810948	1.613585472	15168.32408
13	2.320175578	0.376568896	1.002056599	15132.37573
14	1.946402822	0.330482443	1.168934345	15164.07304
15	1.197268407	0.251574899	0.888189197	15138.46886
16	2.427612827	0.316938932	1.601311207	15044.92709
17	1.928291686	0.351075196	1.082392454	15126.08938



DigiXCT_5_6_1_96

```
In [5]: run "C:\Program Files\Digisens\DigiXCT 5.6.1\demo\bulles.py"
```

Librairie IPSDK

Objectifs de la librairie IPSDK

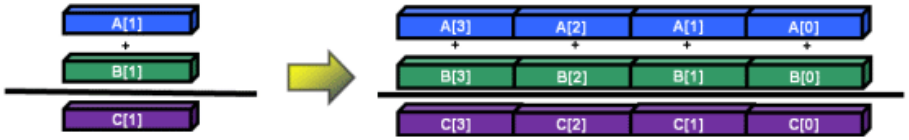
Product Differentiators - IPSDK

- Diversité des algorithmes proposés, implémentation proche de l'état de l'art
- Adaptabilité aux architectures PC (SSE2, AVX2, AVX 512)
- Adaptabilité aux futurs architectures
- Rapidité pour gérer de gros volumes de données (X10 en moyenne)
- Utilisation possible sous forme de batchs
- Stabilité: Outil R&D et industriel.
- Simplicité d'utilisation (Python, C++)
- Documentation complète et précise
- Support technique efficace

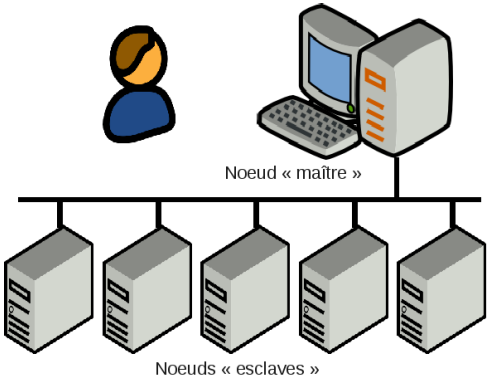
Solutions techniques adoptées

Adaptation à l'architecture PC disponible

- Vectorisation
  - Utilisation systématique des jeux d'instruction SSE, AVX



- Distribution
  - Distribution Multi Core
  - Distribution Multi PC (Cluster)



Core Intellectual Property

- **Ensemble complet de fonctions de traitement d'images (2d/3d/sequence)**
  - Morphology (propagation, erosion, dilatation, closing, …)
  - Filtrage (median, gaussian, bilateral, anisotropic diffusion, NLM, VSNR, …)
  - K-Means, K-means masqué, supervisé ou non
  - Seuillage (otsu, kapur, iso, tophat, hystérésis, …)
  - Statistiques, mesures globales, combinaisons binaires, transformations couleurs, transformation des niveaux de gris, …
  - Connected components, carte de distance, correlation, transformée de hough, analyse de blobs, séparation
  - Recalage par points d'intérêt (Sift)

- **Mise à disposition d'un SDK**
  - C++, python, Windows/Linux
  - Documentation, Wizard

Benchmarks

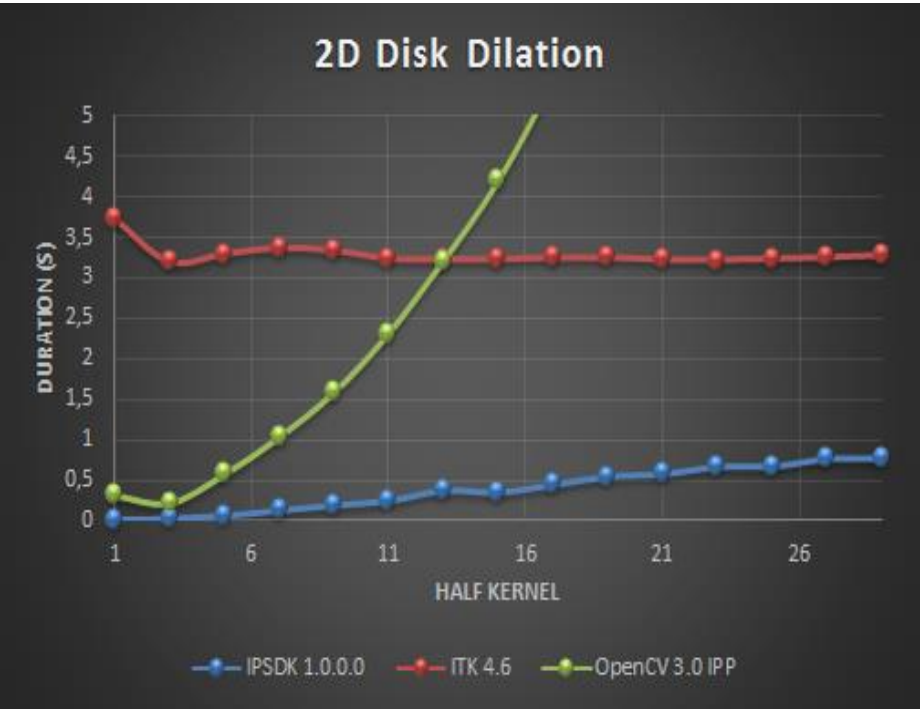


Image 10 000 x 10 000

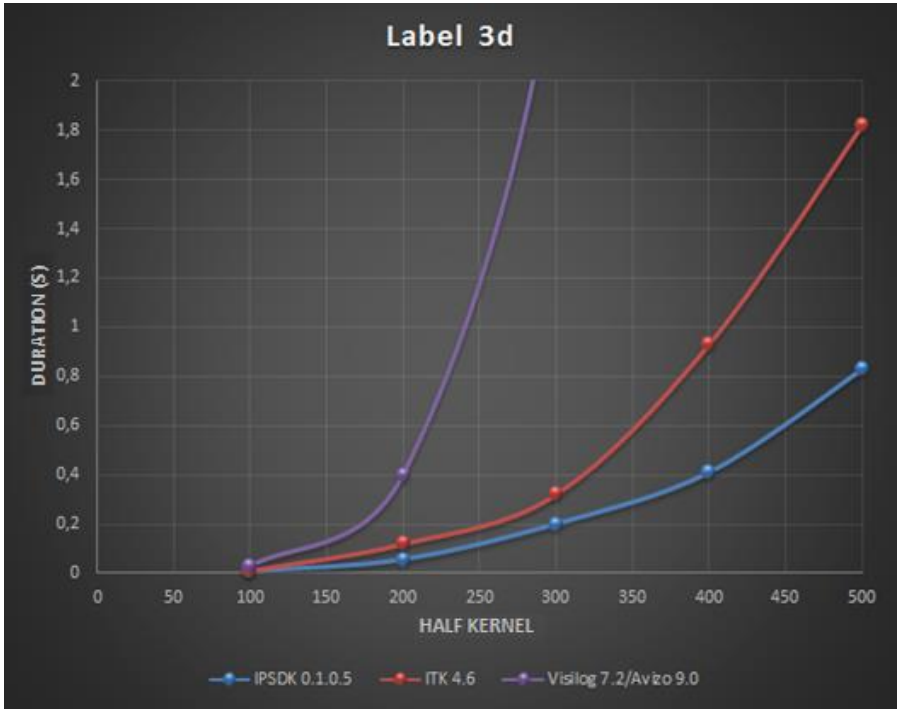


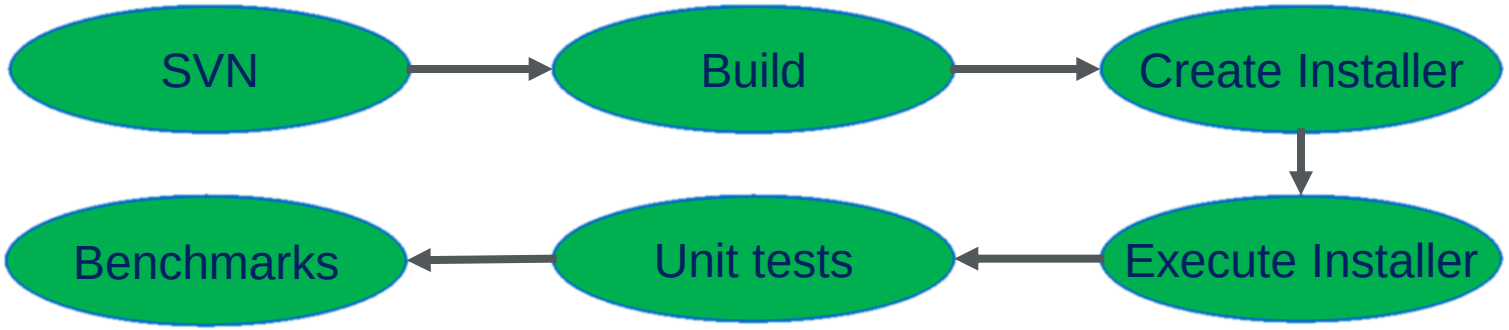
Image 1000 x 1000 x 1000

Projet IPSDK - Tests

Environnement complet de test :

- ✓ Non régression
- ✓ Tests unitaires pour combinaison d'images de tout type
- ✓ Tests de performance (vitesse)
- ✓ Simple à utiliser (mis à disposition dans le SDK)

Nightly build:



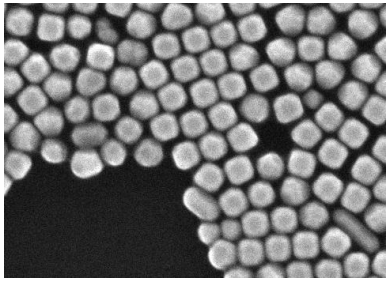
IPSDK
Smart Image Processing

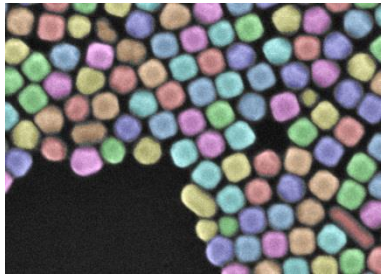
Reactiv'IP
SMART IMAGE PROCESSING

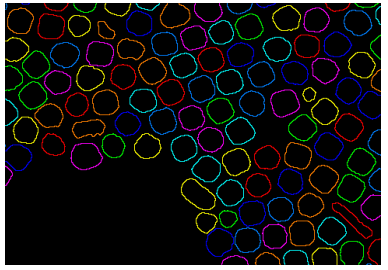
Demonstration

Démonstration

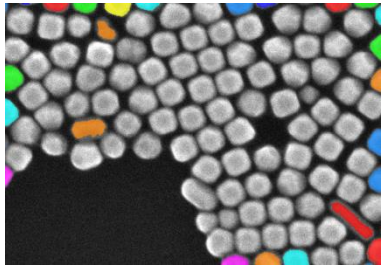
```
# import of IPSDK library
import PyIPSDK
import PyIPSDK.IPSDKIPLAdvancedMorphology as advmorpho
import PyIPSDK.IPSDKIPLBinarization as bin
import PyIPSDK.IPSDKIPLShapeAnalysis as shapeanalysis

# opening of input images
inGreyImg = PyIPSDK.loadTiffImageFile("G:/Sample/blobs_483x348_UInt8.tif")

# automatic binarization
inBinImg,  thrValue = bin.otsuThresholdImg(inGreyImg)

# connected component analysis
inLabelImg2d = advmorpho.connectedComponent2dImg(inBinImg)

# extraction of associated shapes
inShape2dColl = shapeanalysis.labelContourExtraction2d(inLabelImg2d)

# definition of proceeded measure
inMsrInfoSet = PyIPSDK.createMeasureInfoSet2d()
PyIPSDK.createMeasureInfo(inMsrInfoSet, "Circularity2dMsr")

# shape analysis
outMeasureSet = shapeanalysis.shapeAnalysis2d(inGreyImg,  inShape2dColl,  inMsrInfoSet)

# shape filtering computation
outImg = shapeanalysis.shapeFiltering2dImg(inLabelImg2d,  inGreyImg,  "Circularity2dMsr < 0.9")
```







Label Index	Circularity2d	Area2d	Perimeter2d
1	0,73	19,5	21,3
2	0,65	20,0	24,5
3	0,77	260,0	74,6
4	0,82	300,5	74,8
5	0,85	447,0	88,6
6	0,74	220,0	70,8
...	...	...	...
111	0,86	435,0	85,7
112	0,78	339,5	83,2
113	0,59	6,5	15,4



Démonstration

Documentation :

- ✓ Core of library
- ✓ Algorithms

Modules

Here is a list of all modules:

▼ Image processing algorithms	Image processing algorithr
▶ Advanced Morphological image operations	Modules containing advan
▶ Arithmetic image operations	Modules containing arithm
▶ Binarization image operations	Modules containing binariz
▶ Color image operations	Modules containing color c
▼ Image features detection operations	Modules containing featur
Local Extrema Extraction	Finds the local extrema in
Hough Circle Detection	Detects circles in image us
Gradient based Hough Circle Detection	Computes the accumulato
Complex Hough Circle Detection	Computes the complex acc method)
▶ Filtering image operations	Modules containing filterin
▶ Global measure image operations	Modules containing global
▶ Image intensity transformations	Modules containing intens
▶ Logical image operations	Modules containing bitwise
▶ Basic Morphological image operations	Modules containing basic i
▶ Shape analysis	Modules containing shape
▶ Statistic on images	Modules containing image
▶ Utility image processing algorithms	Utility image processing al
▼ Image processing attributes	Modules containing attribu
▶ Attributes	Modules containing attribu
▶ Data item elements	Modules containing data it
▼ Shape Analysis and Measurement	Modules containing shape
▶ Formula	Formula measures associ
▶ Geometry	Modules containing shape
▶ Intensity	Modules containing shape

Detailed Description

algorithm allowing to extract dilated local extrema 2d from an image

This algorithm allows, given an input image *inImg*, a dilation factor *inDilateFactor* and a searched extrema type defined by *inLocalExtremumType* parameter, to compute a binary ouput image *OutBinImg* where all set pixels are part of a dilated local extrema.

Note

This algorithm is an extension of Local Extrema 2d algorithm allowing to dilate (and in some case merge) local extrema. Using this algorithm with a value of *inDilateFactor* set to 0 is equivalent of using Local Extrema 2d algorithm.

Dilation factor *inDilateFactor* is a grey scale factor (not a geometric factor) since local extrema are dilated with a distance based on grey scale difference.

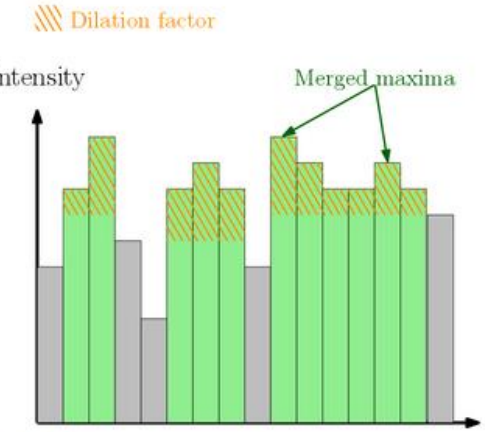
Output image is computed has follow (local maxima case) :

$$OutBinImg[i] = \begin{cases} 1 & if \quad InImg[i] >= InImg[E_i] - InDilateFactor \\ 0 & otherwise \end{cases}$$

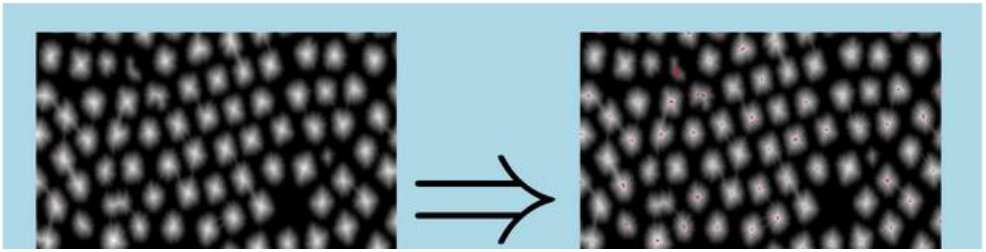
where pixel with index E_i is :

- connected to pixel with index i in output image
- greater or equal to all other values (local maxima) in connected component associated pixel with index i .

This is illustrated in case of a 1d signal :



Here is an example of a dilated local maxima extraction applied to an unsigned char input image with dilation factor equal to 0 :



Démonstration

Example of Python code :

Example imports

```
import PyIPSDK
import PyIPSDK.IPSDKIPLFiltering as filter
```

Code Example

```
    # opening of input images
    inImg = PyIPSDK.loadTiffImageFile(inputImgPath)

    # median filter 3d computation
    outImg = filter.median3dImg(inImg, 3, 3, 3)
```

Example of C++ code :

Header file

```
#include <IPSDKIPL/IPSDKIPLFiltering/Processor/Median3dImg/Median3dImg.h>
```

Code Example

```
        // opening input image
        ImagePtr pInImg = loadTiffImageFile(inputImgPath);

        if(pInImg->getBufferType() != inImageBufferType)
            pInImg = util::convertImg(pInImg, inImageBufferType);

        ImageGeometryPtr pOutImageGeometry =
            geometry3d(outImageBufferType, pInImg->getSizeX(), pInImg->getSizeY(), pInImg->getSizeZ());

        boost::shared_ptr<MemoryImage> pOutImg =
            boost::make_shared<MemoryImage>();
        pOutImg->init(*pOutImageGeometry);

        // compute median3d of input image
        median3dImg(pInImg, inHalfKnlSizeX, inHalfKnlSizeY, inHalfKnlSizeZ, pOutImg);
```

Démonstration

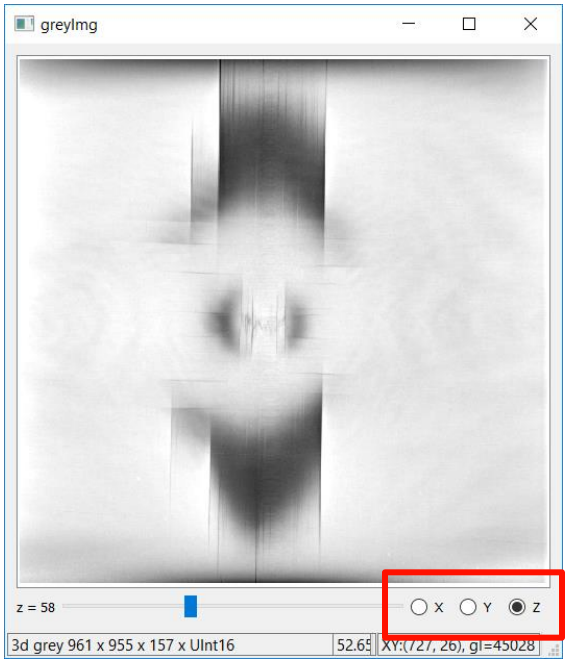
Utilisation d'IPSDK avec des « third parties »

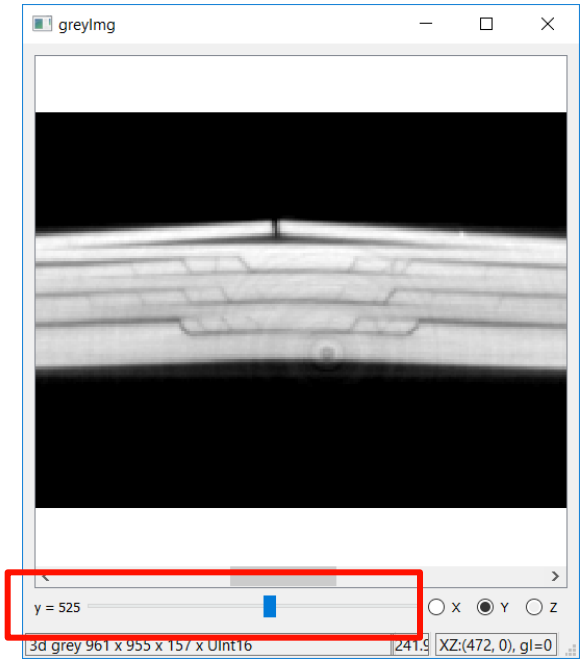
```
import os
import PyIPSDK
import cv2
import numpy as np

# opening of input image
img2d1 = PyIPSDK.loadTiffImageFile("G:/Sample/blobs_483x348_UInt8.tif")

# threshold this image using OpenCV function
threshold1, cvBinImg2d1 = cv2.threshold(img2d1.array, 127, 255, cv2.THRESH_BINARY)
print("Threshold1 = " + str(threshold1))

# once done we can retrieve a PyIPSDK image
# note that cvBinImg2d1 and binImg2d1 share memory
binImg2d1 = PyIPSDK.fromArray(cvBinImg2d1)

# and finaly save resulting image
PyIPSDK.saveTiffImageFile("G:/Sample/PyIPSDK_mix_with_OpenCV_1.tif", binImg2d1)
```

Outils graphiques d'IPSDK

Visualisation d'images

```
import PyIPSDK.IPSDKUI as gui

def main(argv):

    inputImgPath = "D:/Data/Toulouse/SlicesZ.tif"

    # opening of input image
    greyImg = PyIPSDK.loadTiffImageFile( inputImgPath, PyIPSDK.eTiffDirectoryMode.eTDM_Volume)
    gui.displayImg(greyImg, 'greyImg')
```

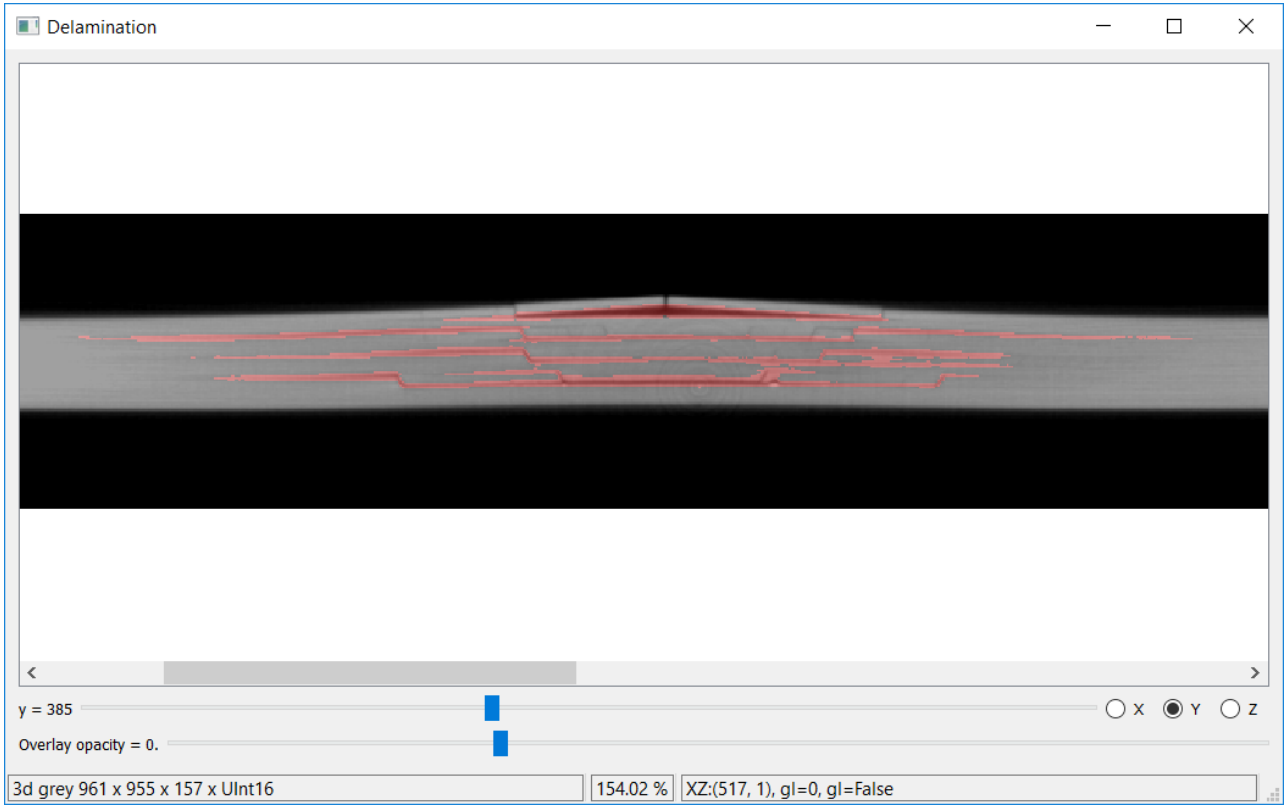




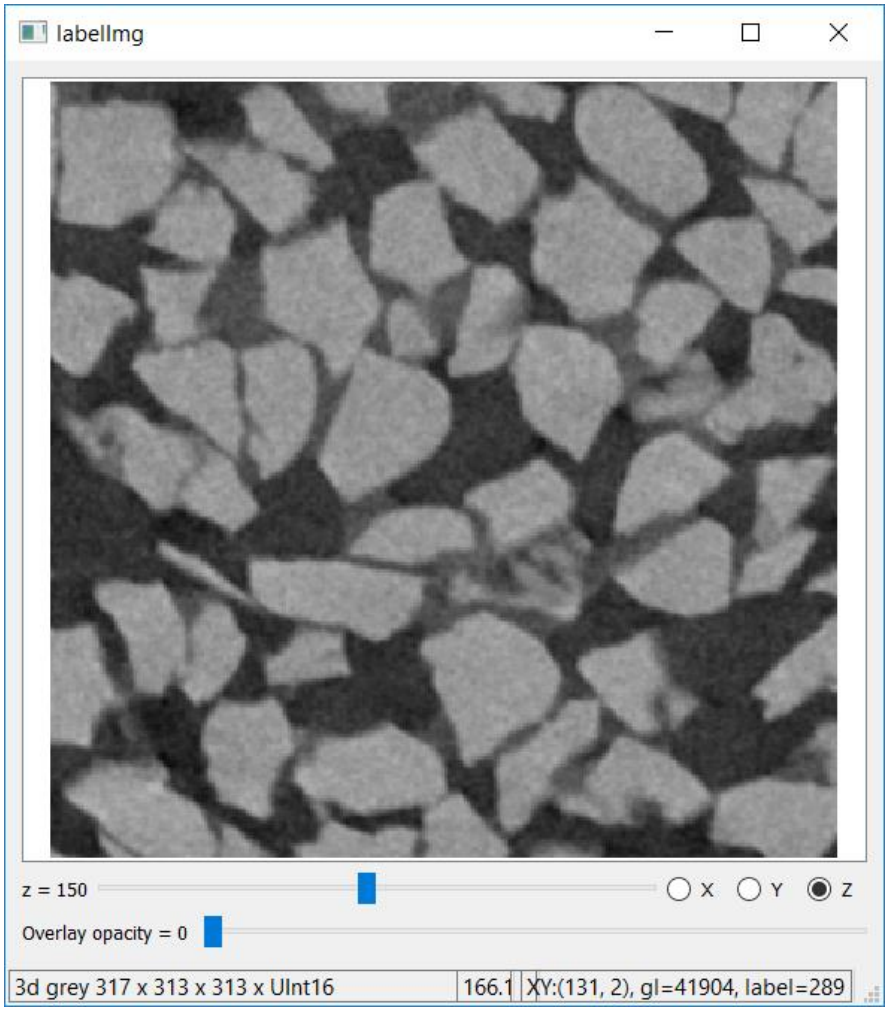
Outils graphiques d'IPSDK

Visualisation d'images + overlay

```
import PyIPSDK.IPSDKUI as gui

def main(argv):

    gui.displayImg(greyImg, 'Delamination',biggest)
```



Outils graphiques d'IPSDK



Outils d'échange

Interaction avec DragonFly, Digi-XCT ou Avizo

```
import sys
import PyIPSDK
import PyIPSDK.IPSDKIPLFiltering as filter

# Communication with Digi-XCT
def DigiXCTMacro():
    greyImg = PyIPSDK.fromDigisens('bloc3d')

    greyFilteredImg = filter.separatedBilateral3dImg(greyImg, 3,8)

    PyIPSDK.toDigisens(greyFilteredImg, 'MyResult', 'C:\MyResult.vol')


# Communication with Avizo
def AvizoMacro():
    greyImg = PyIPSDK.fromAvizo('bloc3d')

    greyFilteredImg = filter.separatedBilateral3dImg(greyImg, 3,8)

    PyIPSDK.toAvizo(greyFilteredImg, 'MyResult')

# Communication with ORS DragonFly
def DragonFlyMacro():
    greyImg = PyIPSDK.fromDragonFly('bloc3d')

    greyFilteredImg = filter.separatedBilateral3dImg(greyImg, 3,8)

    PyIPSDK.toDragonFly(greyFilteredImg, 'MyResult')
```

IPSDK
Smart Image Processing

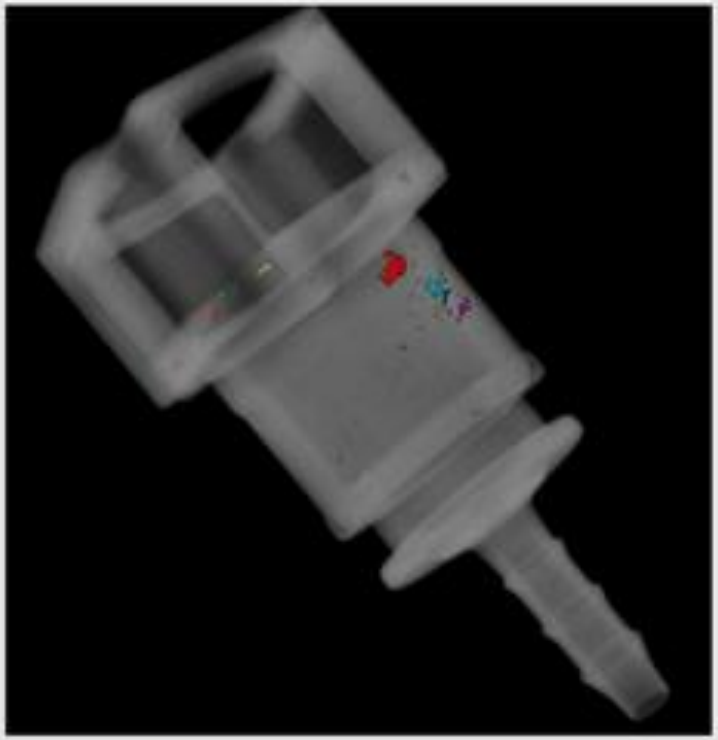
Reactiv'IP
SMART IMAGE PROCESSING

Exemples d'utilisation

Exemples d'utilisation

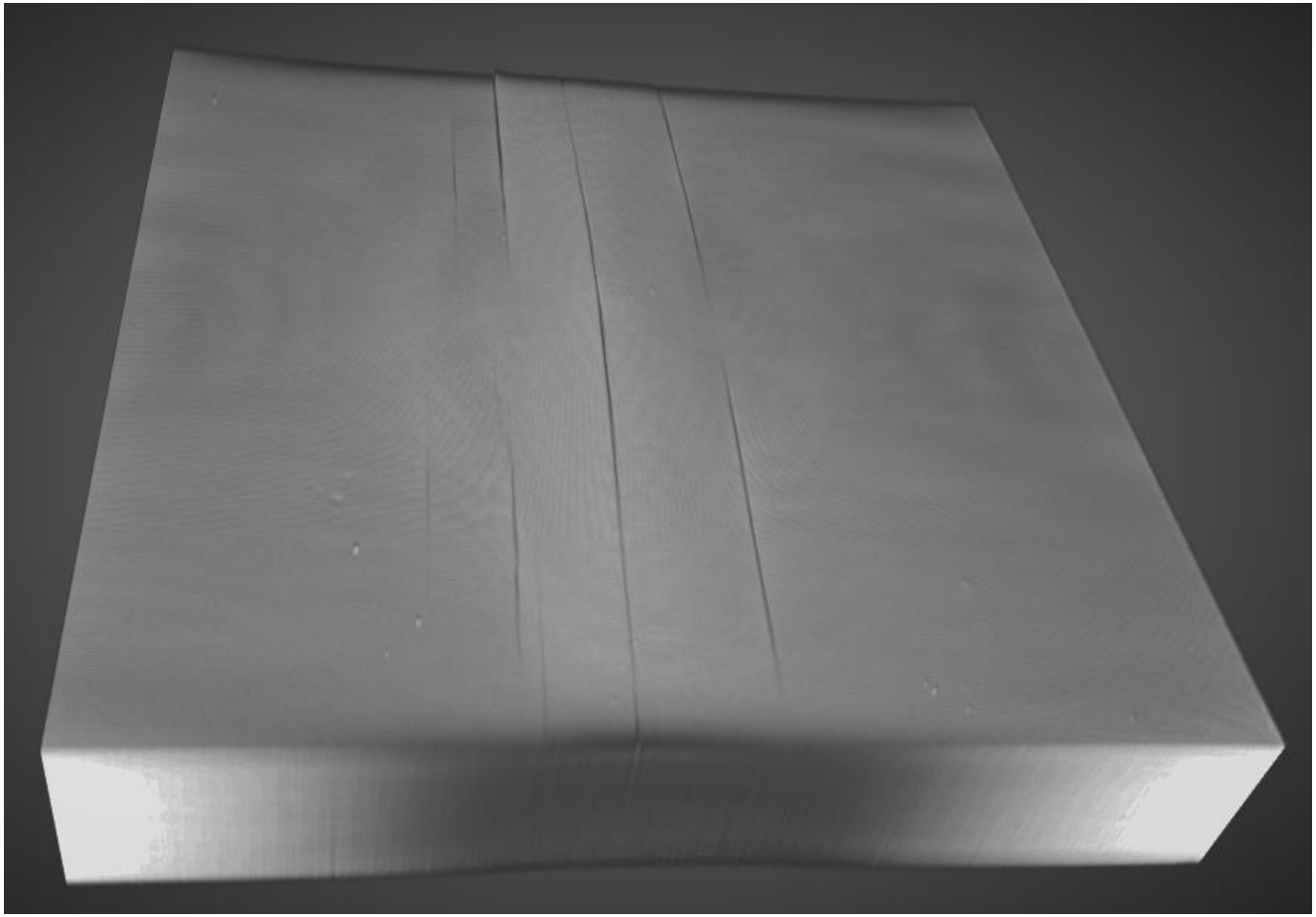
Segmentation de porosités → **Classique, disponible dans tous les softs**

Image originale (exemple 1)

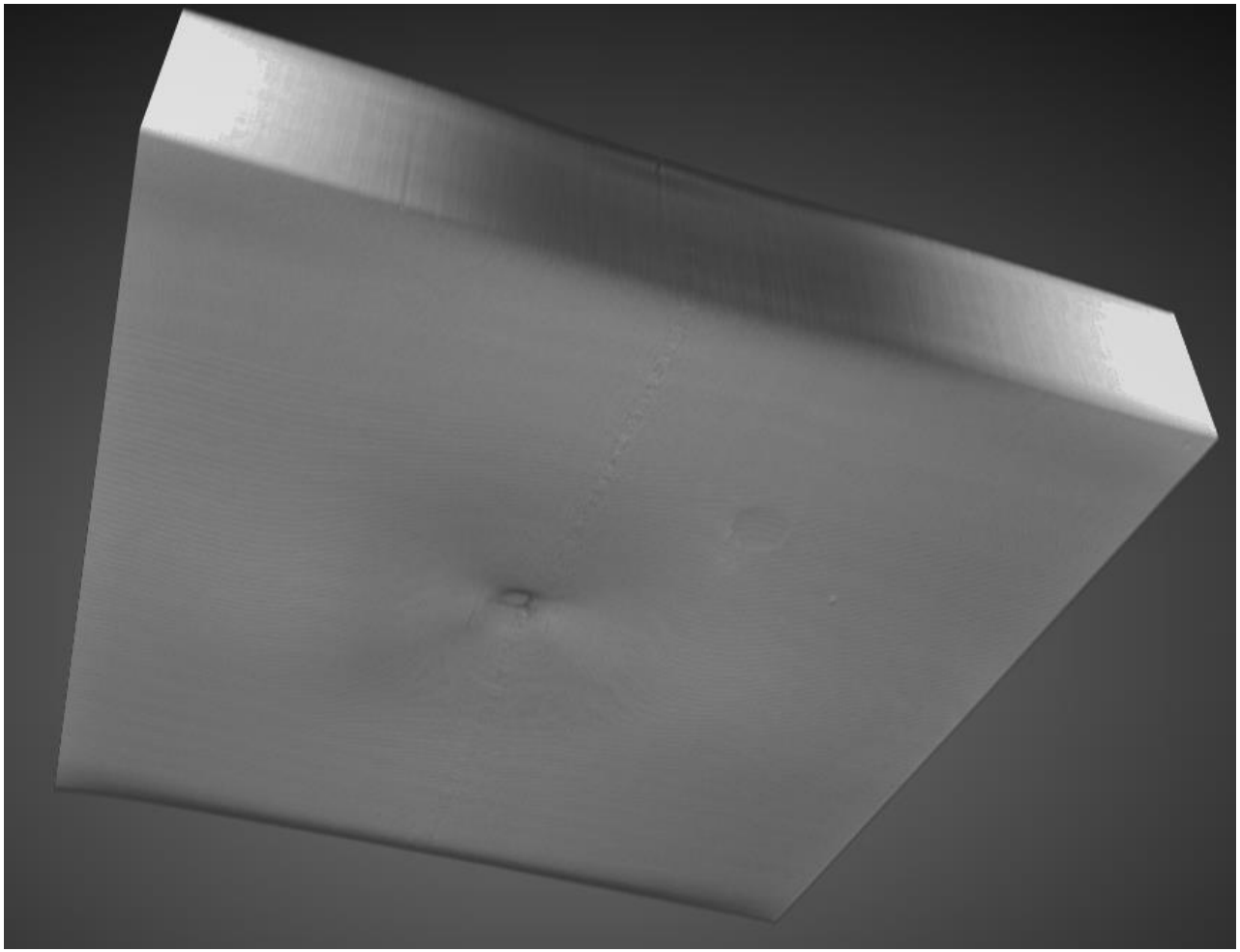


Détection, mesure et visualisation des porosités.

Composite

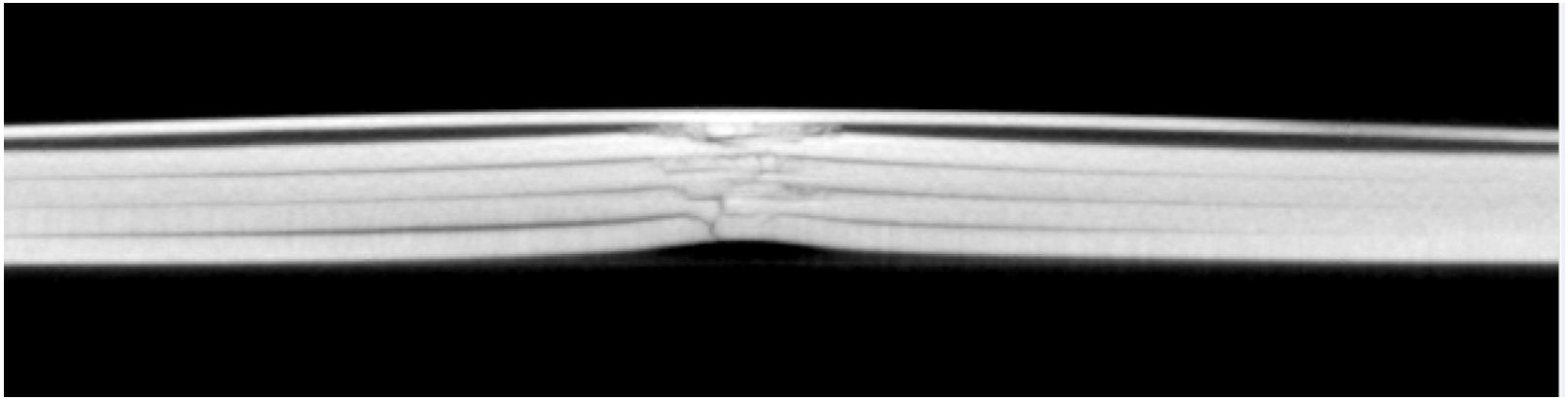


Composite



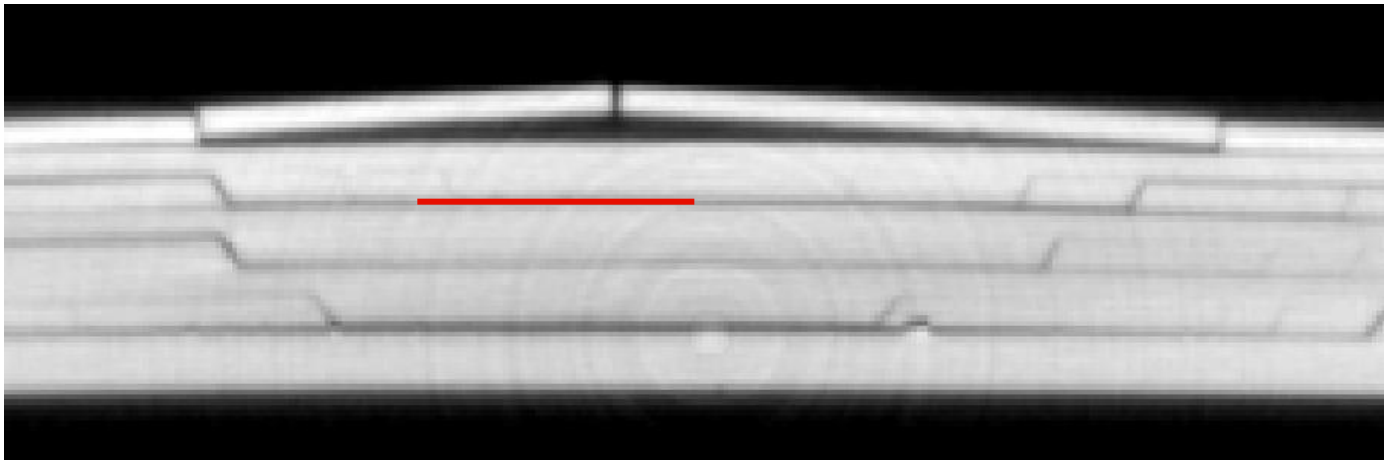
Composite - Seuillage

Image brute:

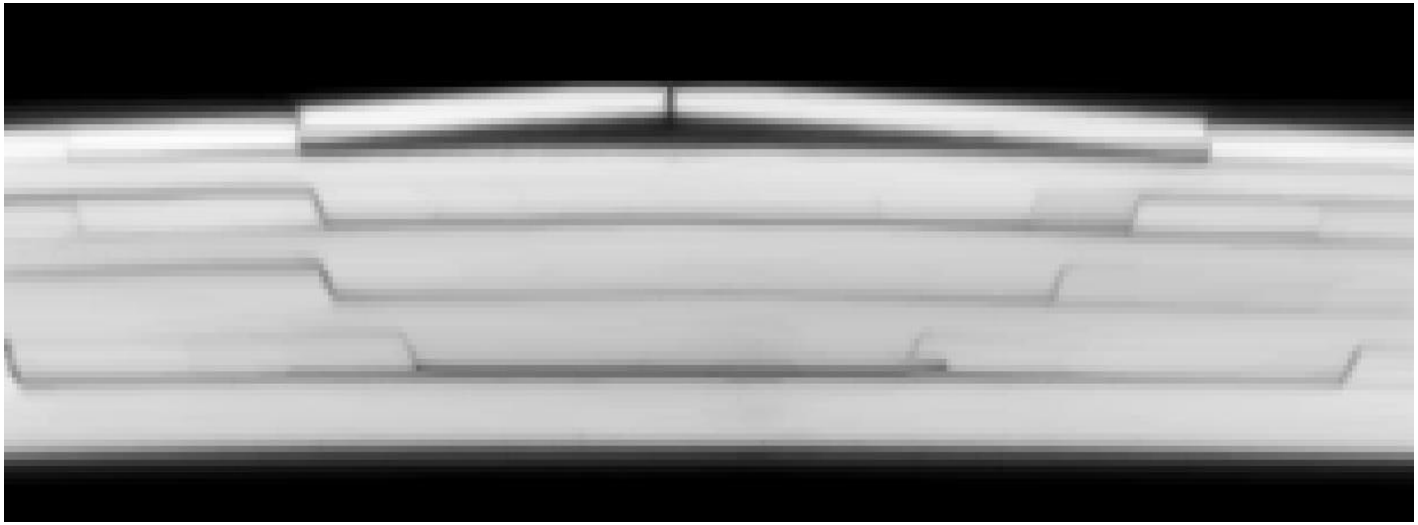


Seuillage impossible en l'état:

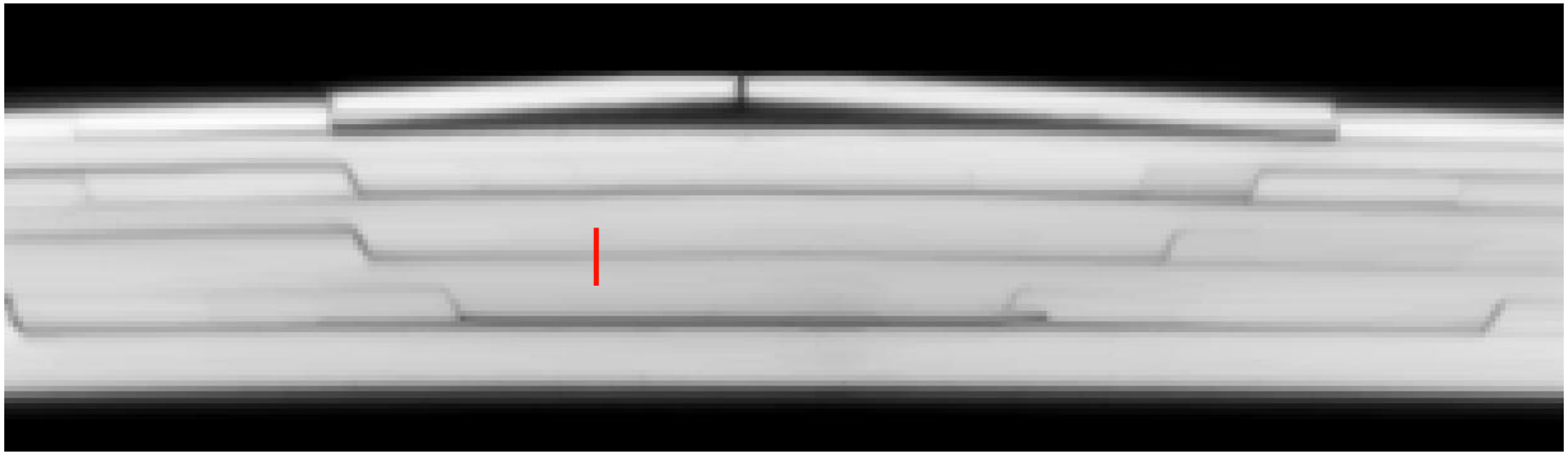
Composite – Morphologie orientée

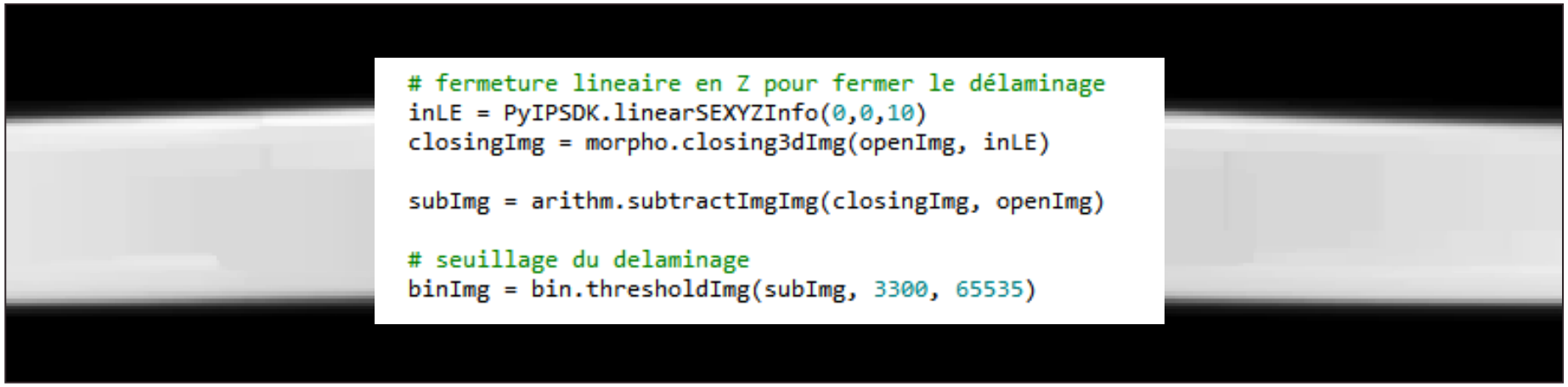


```
 # definition of used structuring element
inSE = PyIPSDK.circularSEXYInfo(11)

# ouverture en XY pour renforcer le délaminage
openImg = morpho.opening2dImg(greyImg, inSE)
```



Composite – Tophat orienté





```
# fermeture lineaire en Z pour fermer le délaminage
inLE = PyIPSDK.linearSEXYZInfo(0,0,10)
closingImg = morpho.closing3dImg(openImg, inLE)

subImg = arithm.subtractImgImg(closingImg, openImg)

# seuillage du delaminage
binImg = bin.thresholdImg(subImg, 3300, 65535)
```

Composite - Seuillage

Image brute:

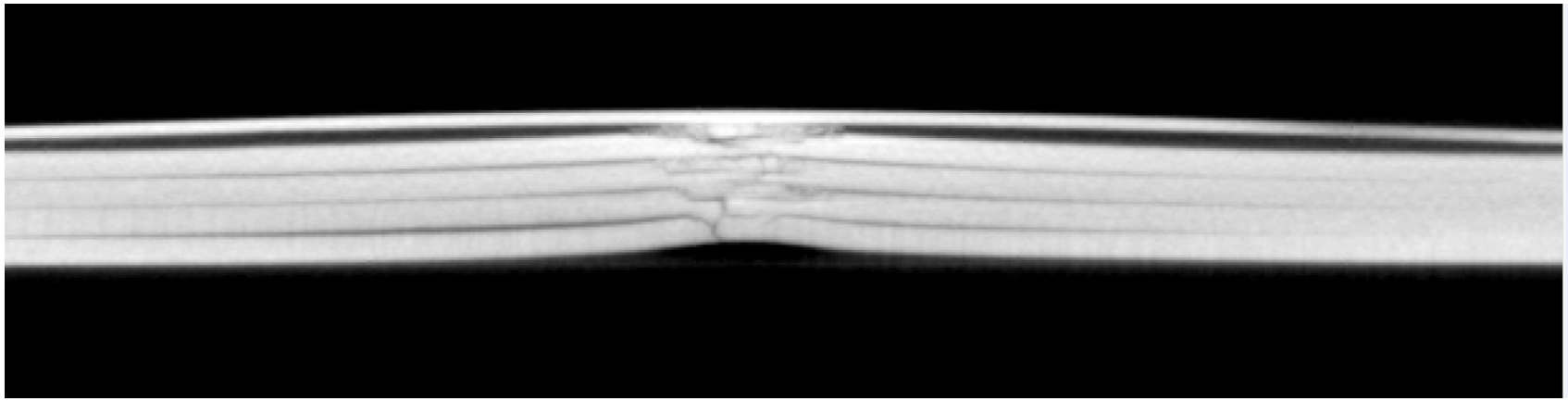
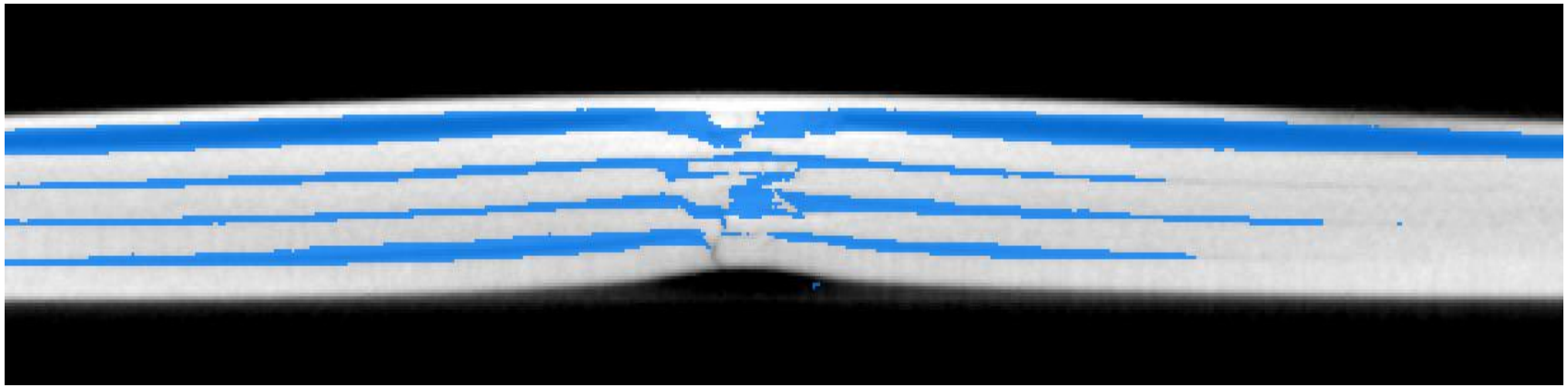
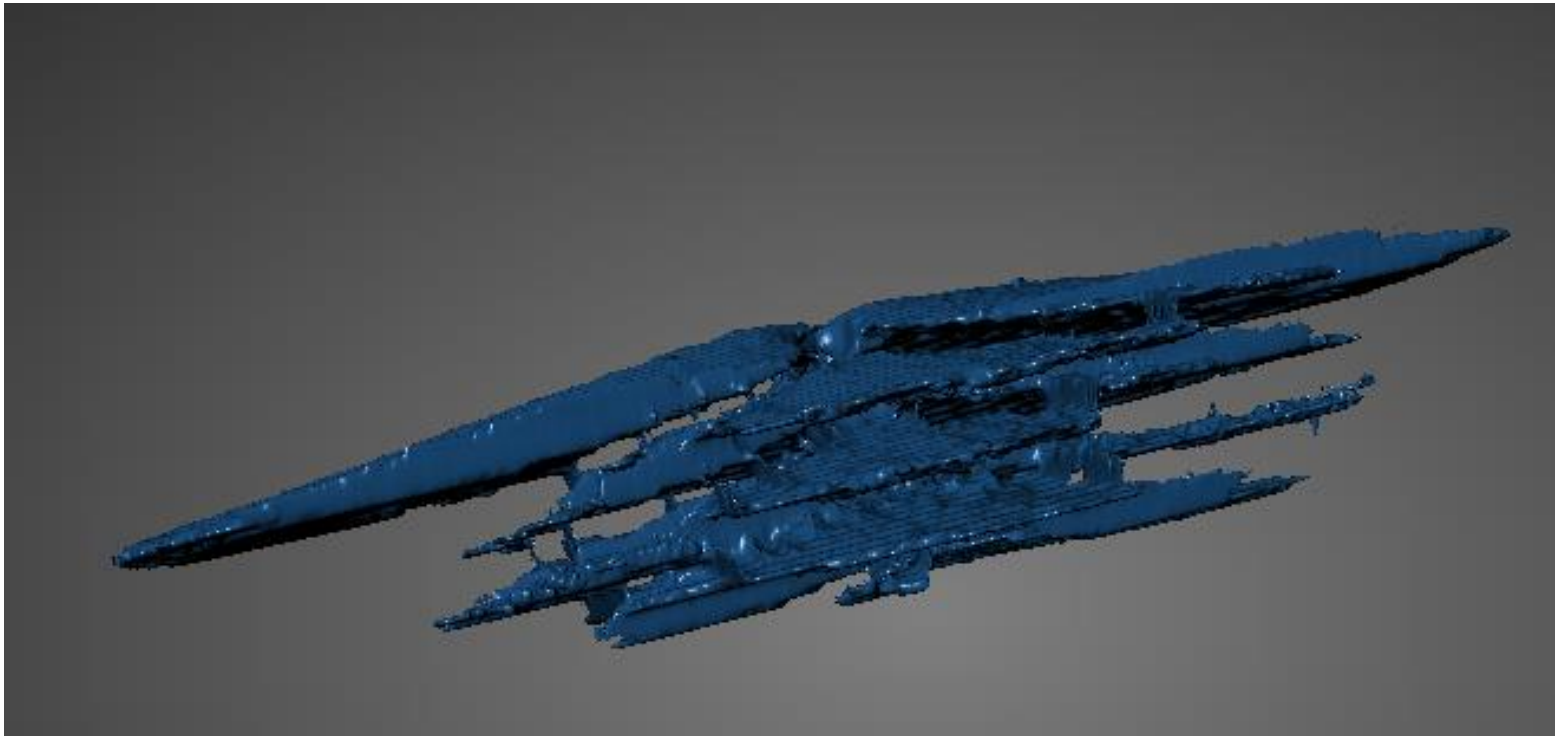


Image filtrée maintenant seuillable (Tophat orienté):



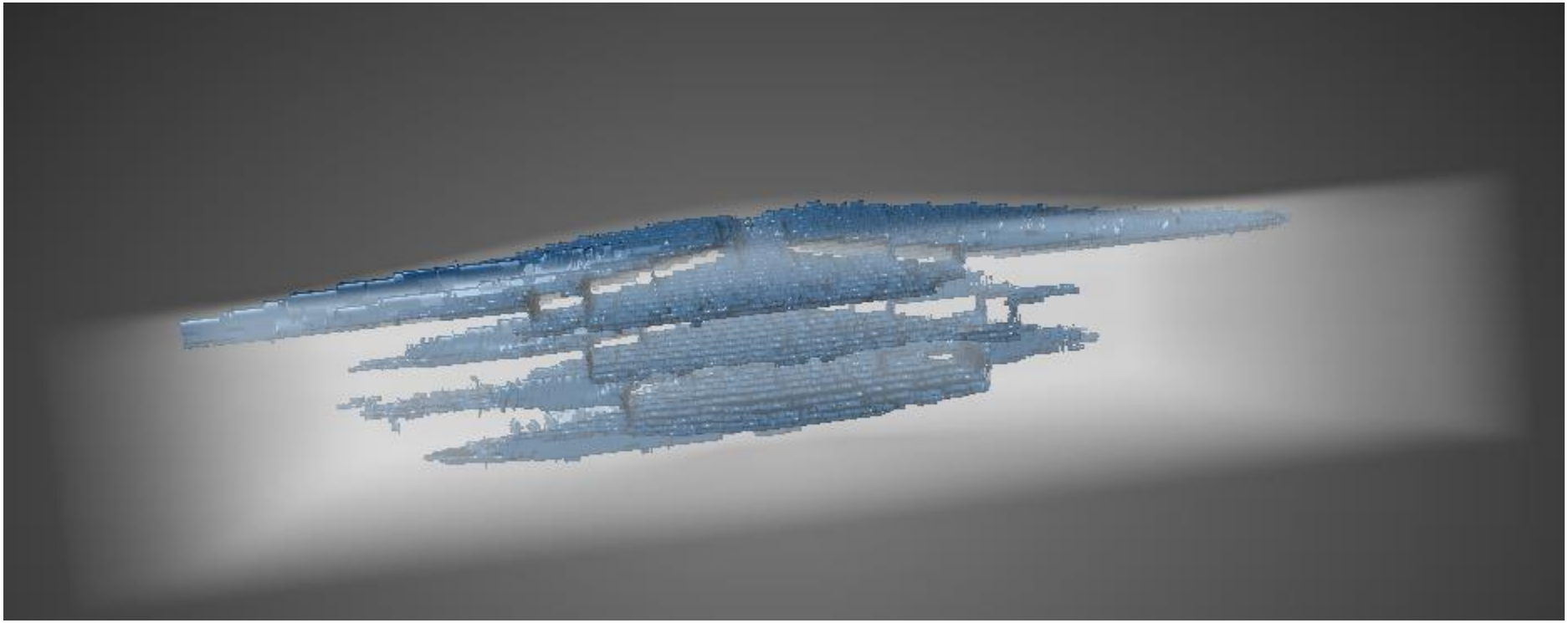
Composite

Reconstruction volumique 3D de la délamination:



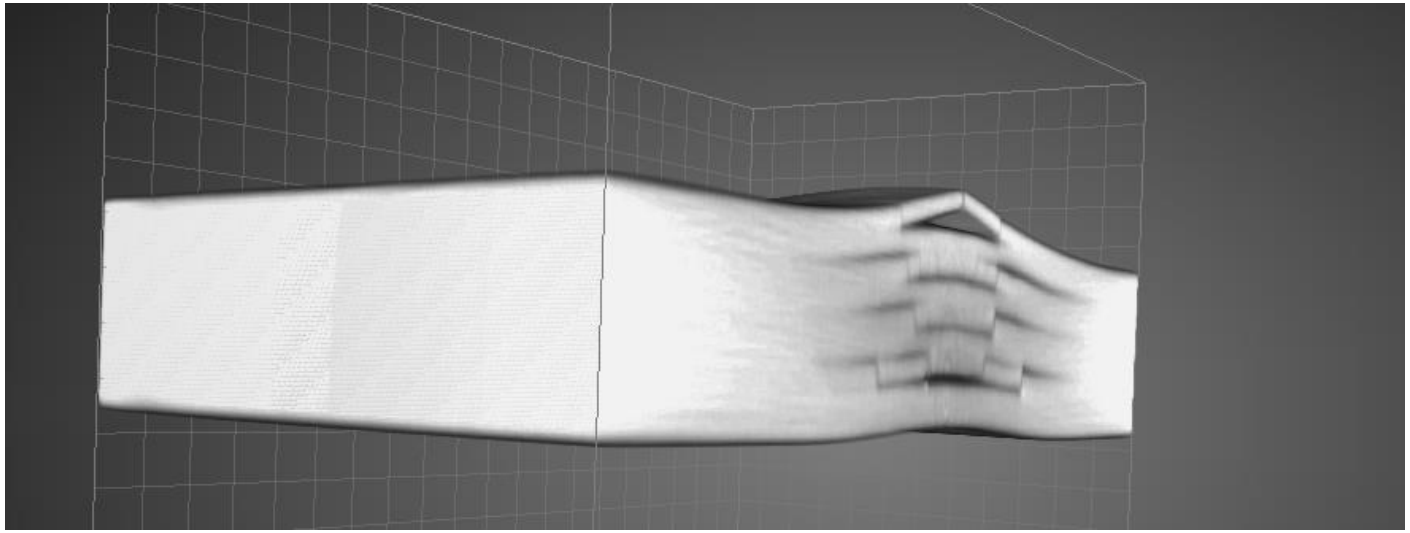
Composite

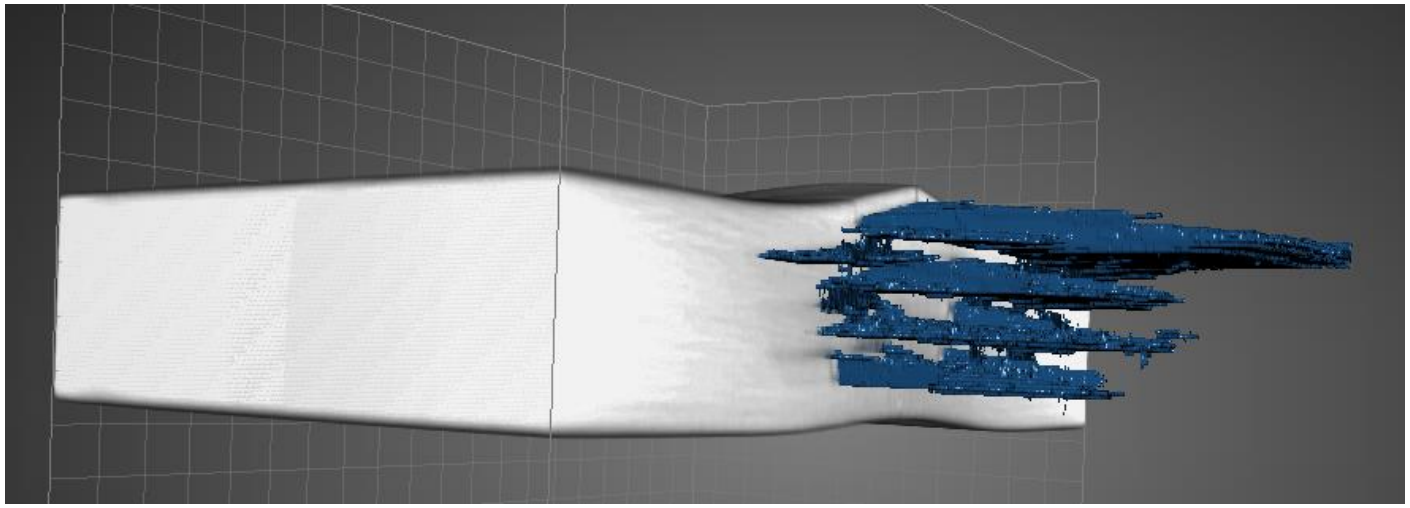
Visualisation 3D dans le volume:



Composite

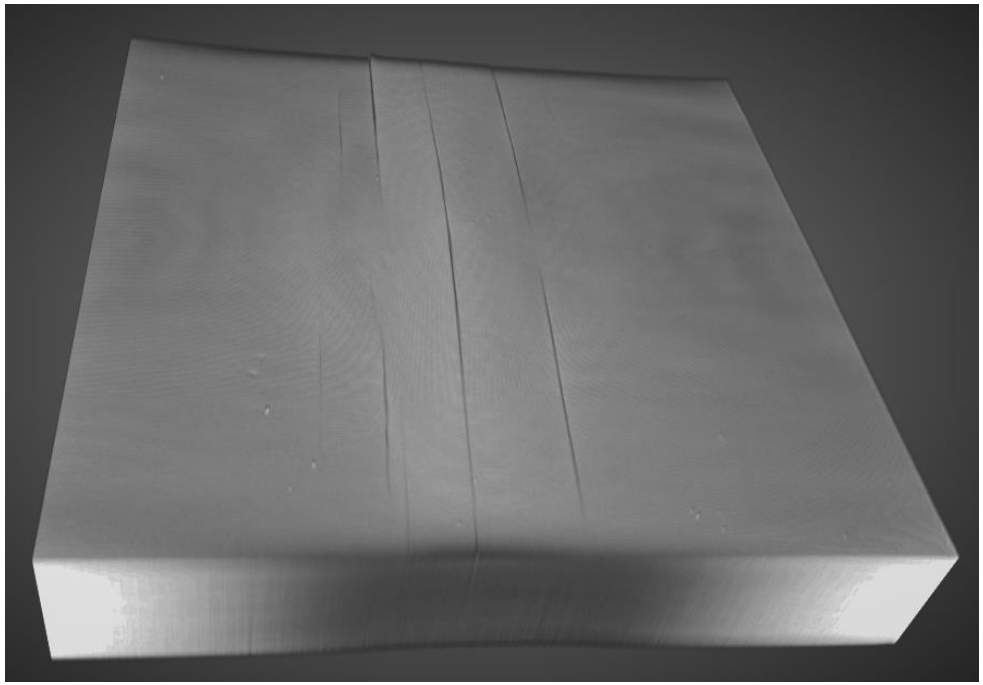
Visualisation 3D dans le volume:

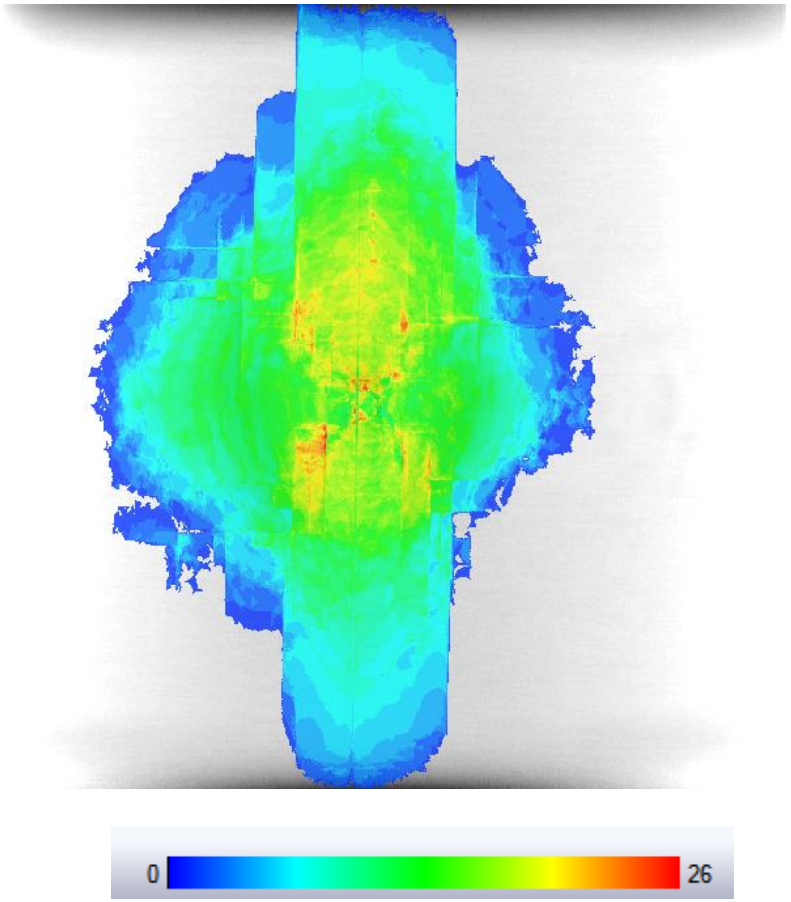




Composite

Cartographie de l'épaisseur cumulée de délamination





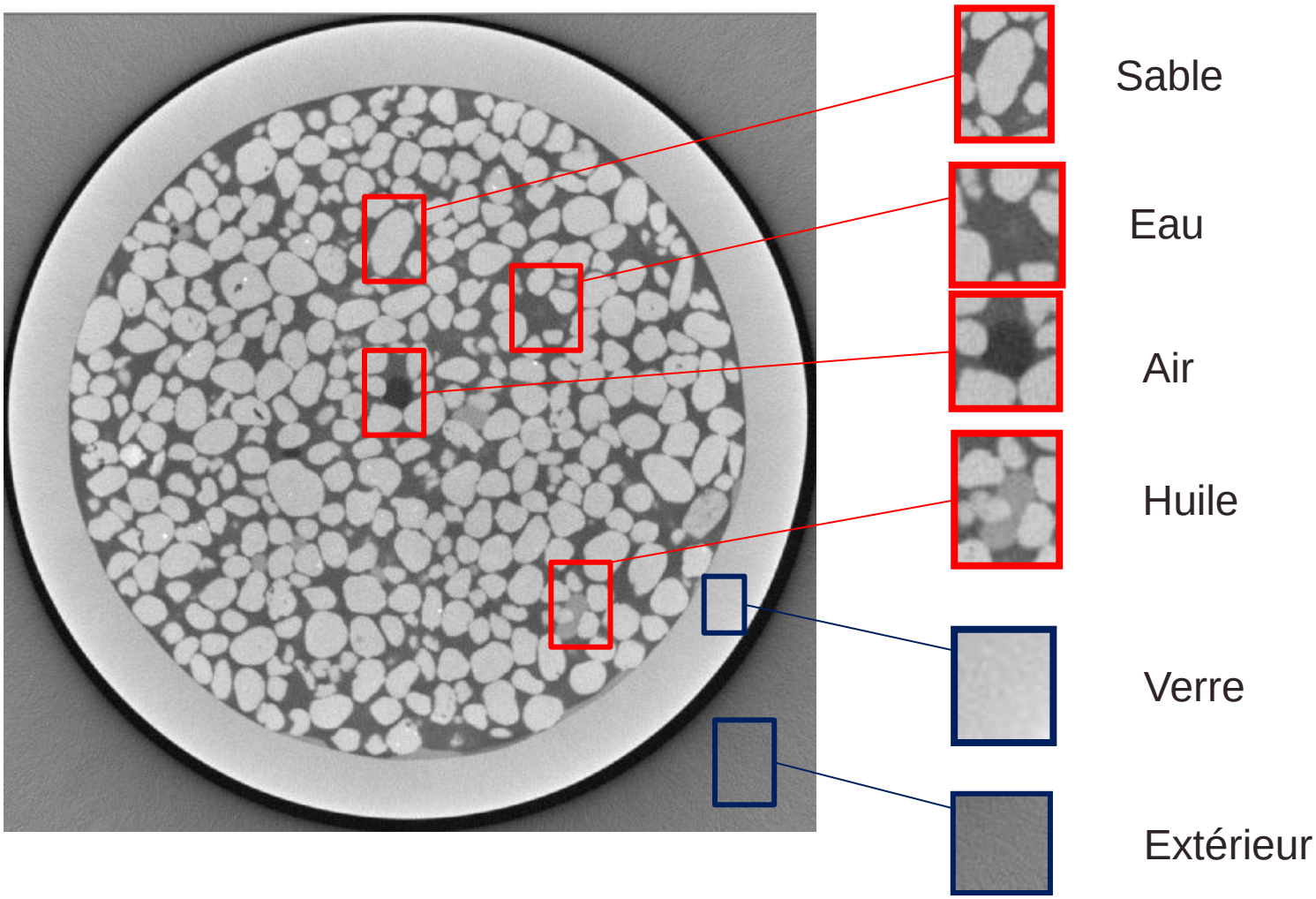
Composite

Script Python correspondant (avec utilisation d'IPSDK)

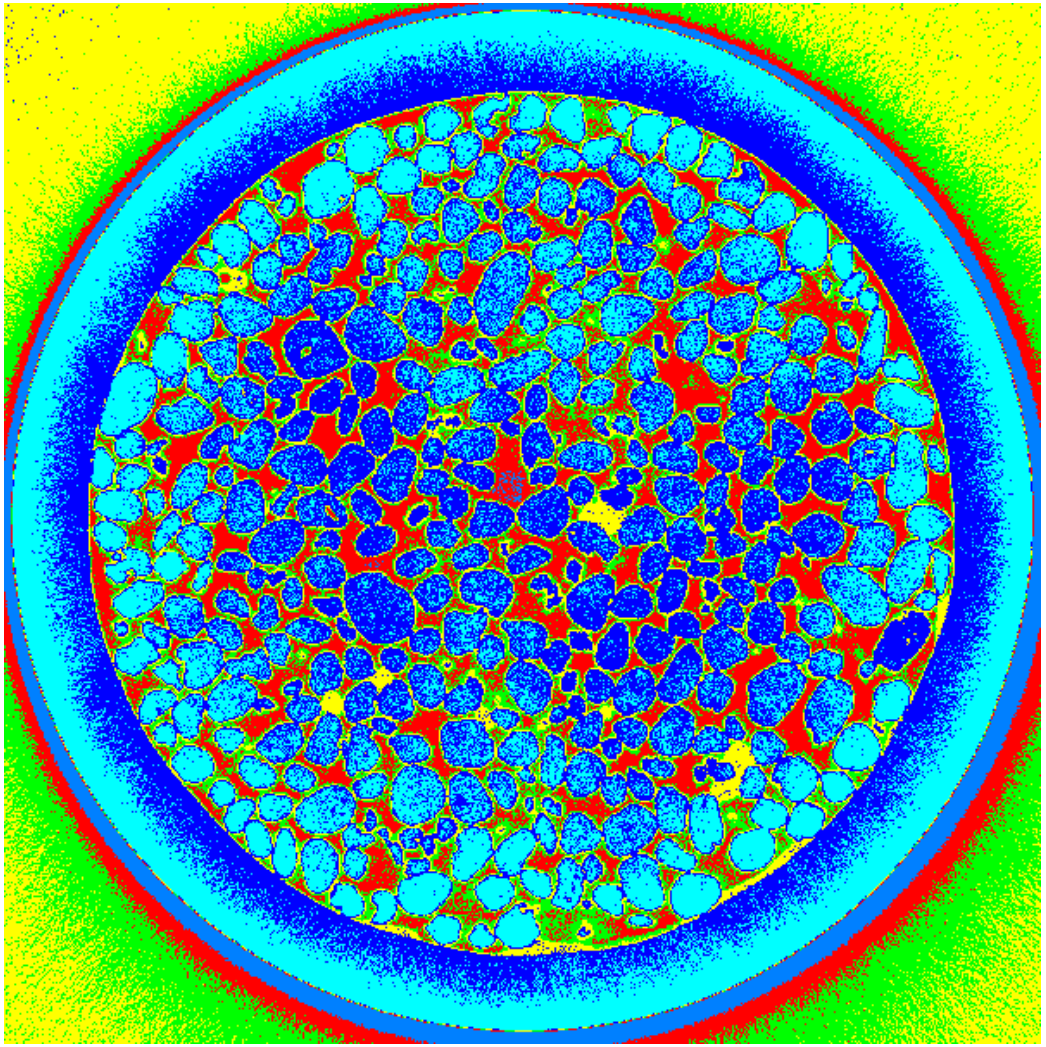
```
# opening of input image
greyImg = PyIPSDK.loadTiffImageFile( inputImgPath, PyIPSDK.eTiffDirectoryMode.eTDM_Volume)

 # definition of used structuring element
inSE = PyIPSDK.circularSEXYInfo(11)

# ouverture en XY pour renforcer le délaminage
openImg = morpho.opening2dImg(greyImg, inSE)

# fermeture lineaire en Z pour fermer le délaminage
inLE = PyIPSDK.linearSEXYZInfo(0,0,10)
closingImg = morpho.closing3dImg(openImg, inLE)

subImg = arithm.subtractImgImg(closingImg, openImg)

# seuillage du delaminage
binImg = bin.thresholdImg(subImg, 3300, 65535)

# nettoyage
inSE = PyIPSDK.sphericalSEXYZInfo(2)
erodeImg = morpho.erode3dImg(binImg, inSE)
binImgClean = advmorpho.binaryReconstruction3dImg(binImg, erodeImg)
biggest = advmorpho.keepBigShape3dImg(binImgClean, 1)

outImg = gblmsr.seqProjectionImg(ImgSeq, PyIPSDK.eProjStatType.ePST_Sum)
gui.displayImg(outImg, 'outImg' )

PyIPSDK.saveTiffImageFile(outputImgPath5, outImg)
```

Exemples d'utilisation

Segmentation et classification multi phases 3D

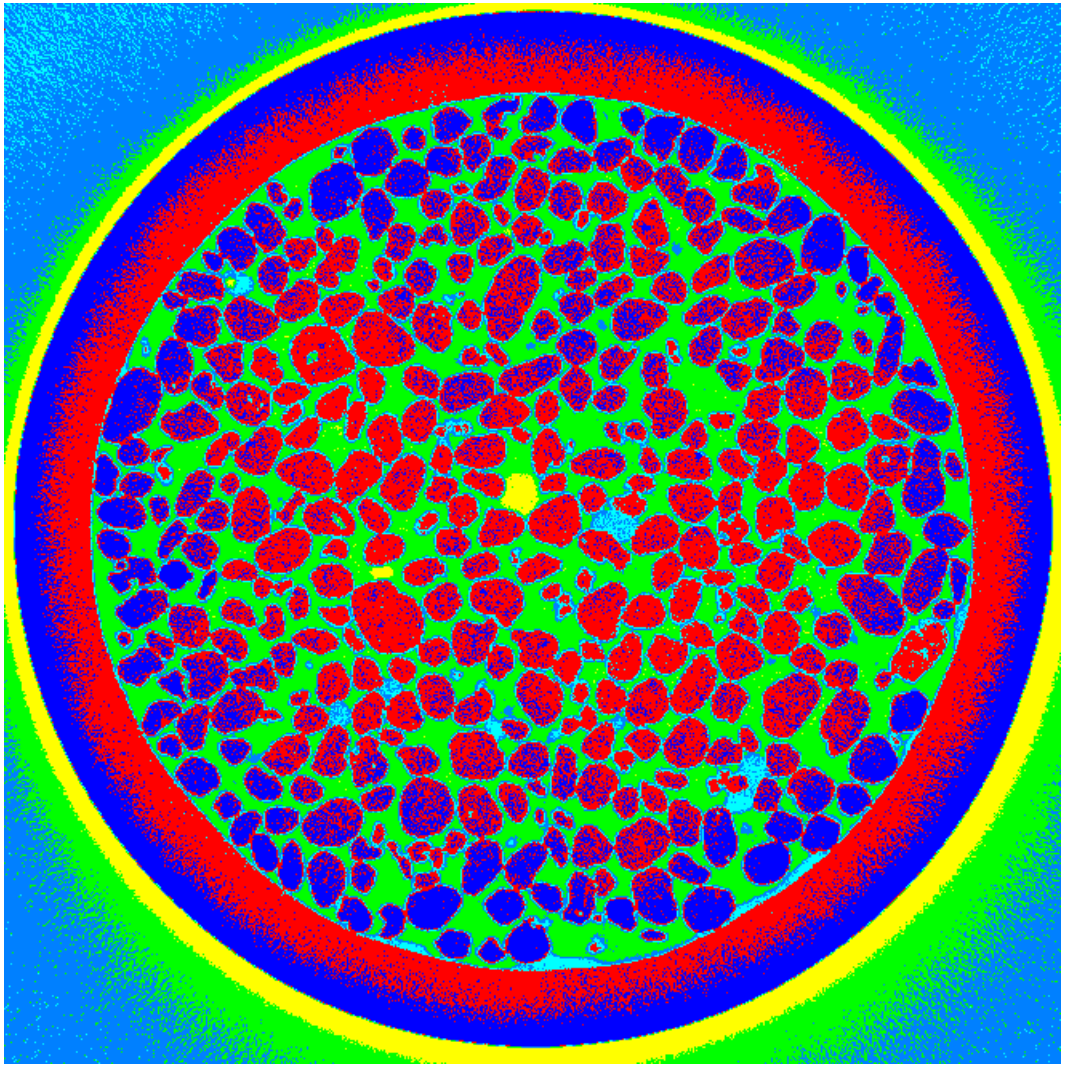


Classification 3D



K-Means automatique
avec 6 classes

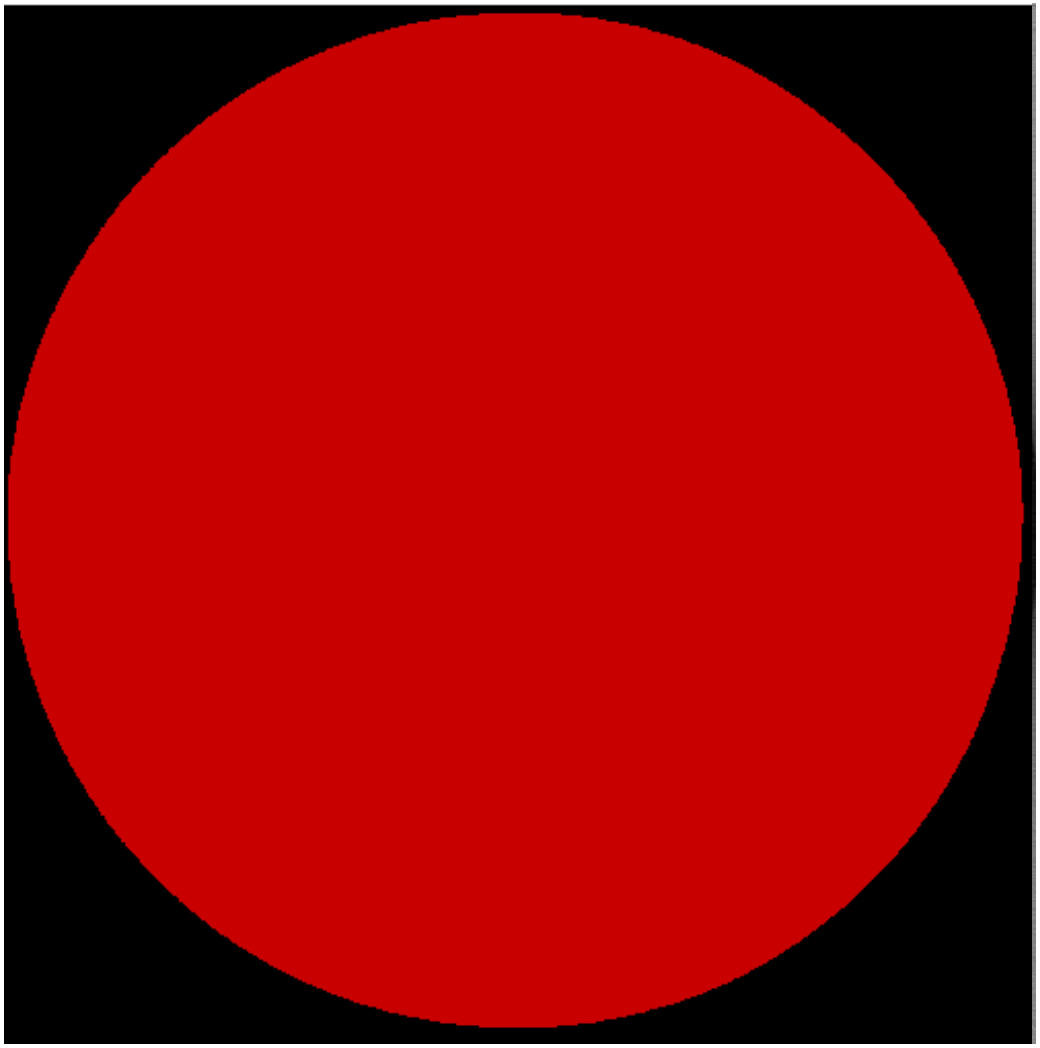
Classification 3D



Approche par apprentissage

Deep learning

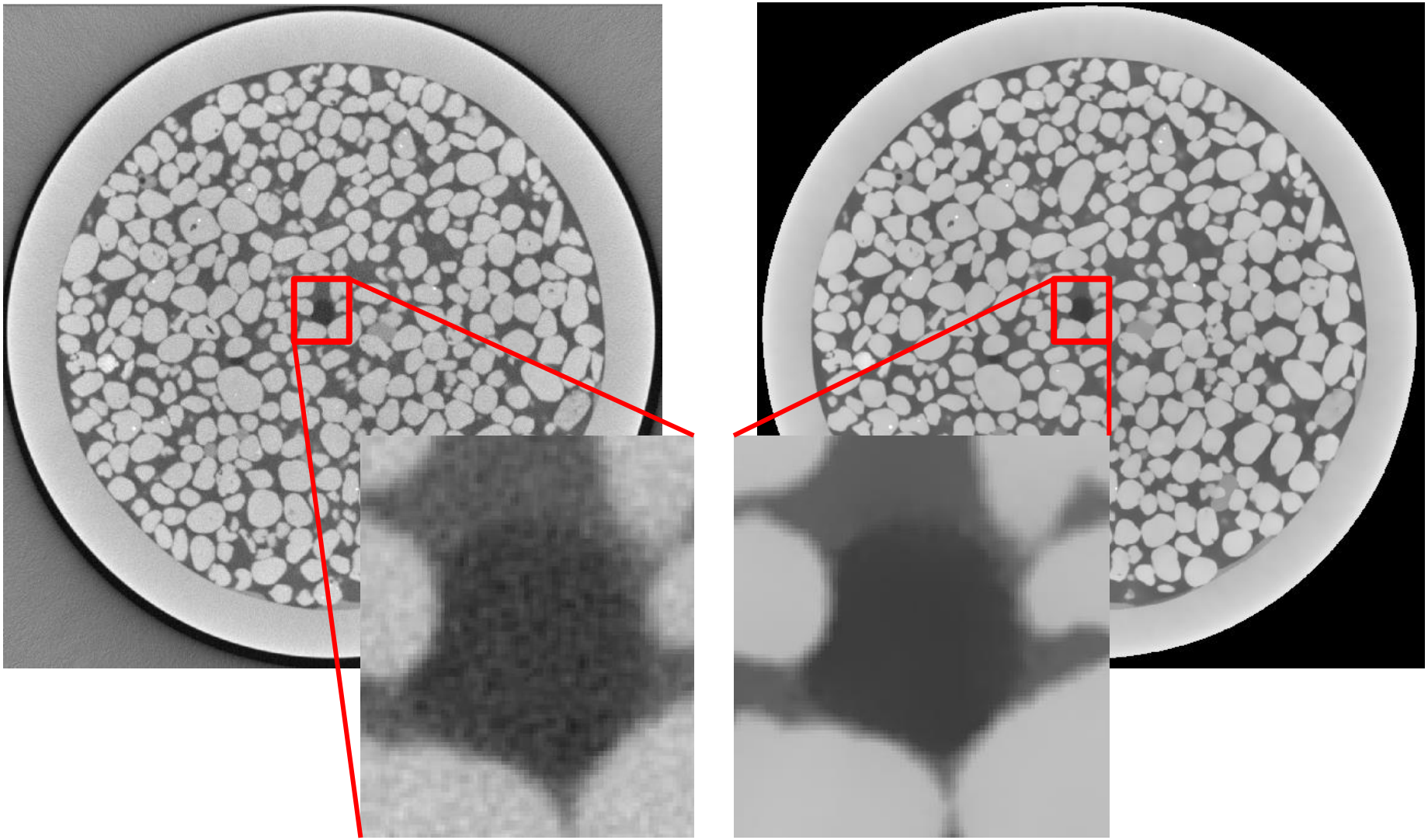
Classification 3D



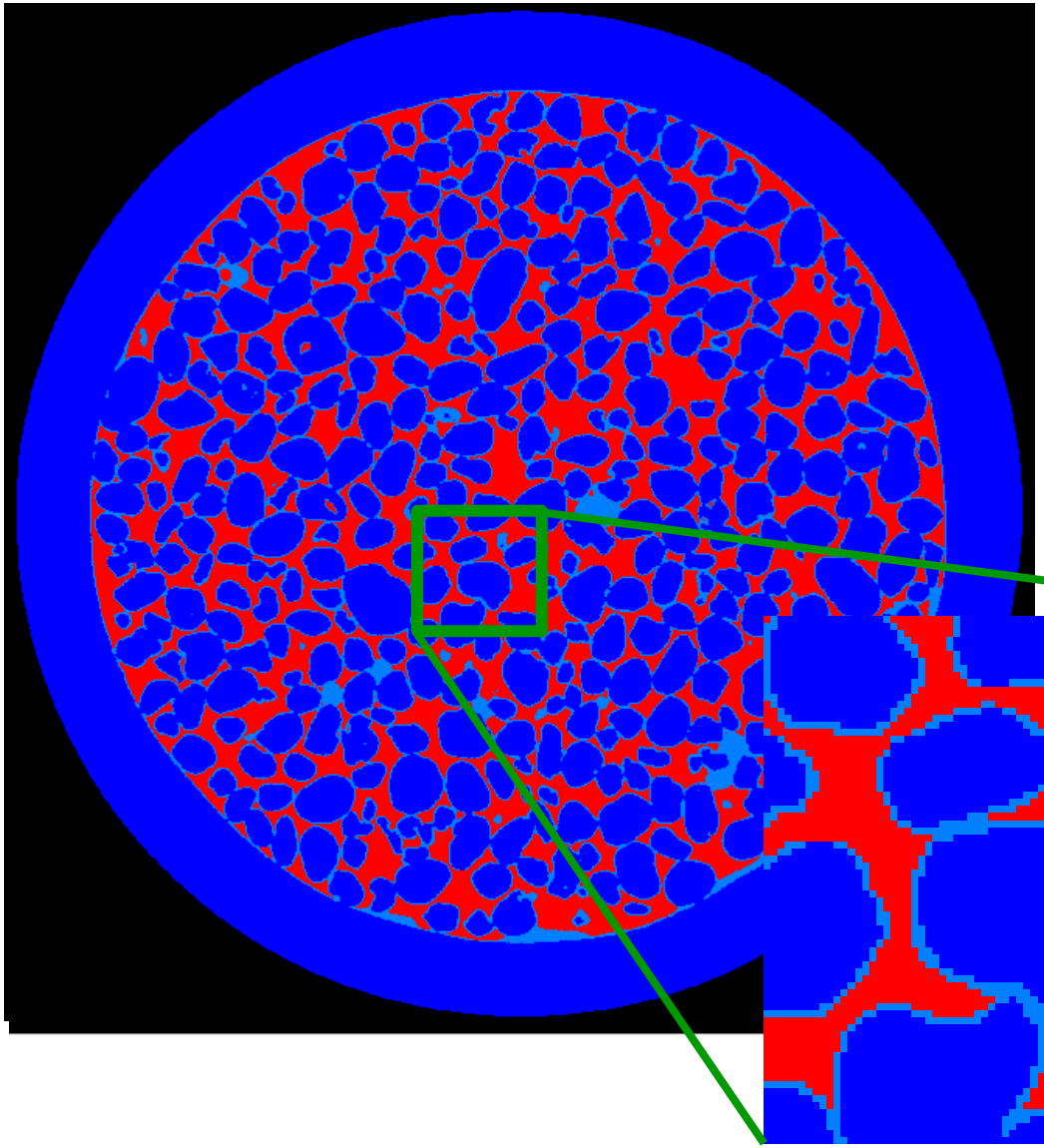
Détection du masque

Classification 3D

Débruitage: Non Local Means



Classification 3D



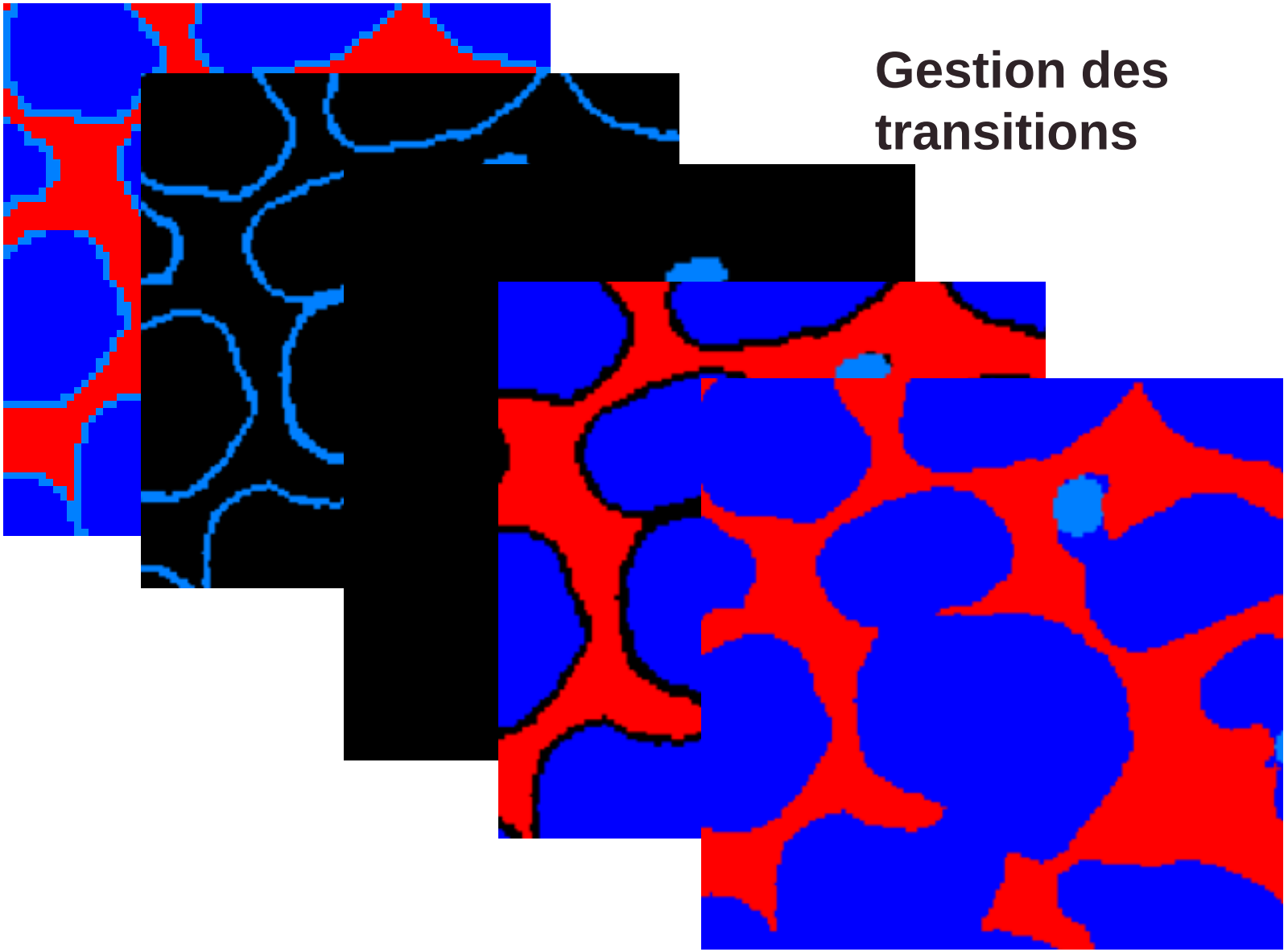
- Utilisation du K-Means masqué
- Eau et air trop proches

  Première étape:

  Classification en 3 classes

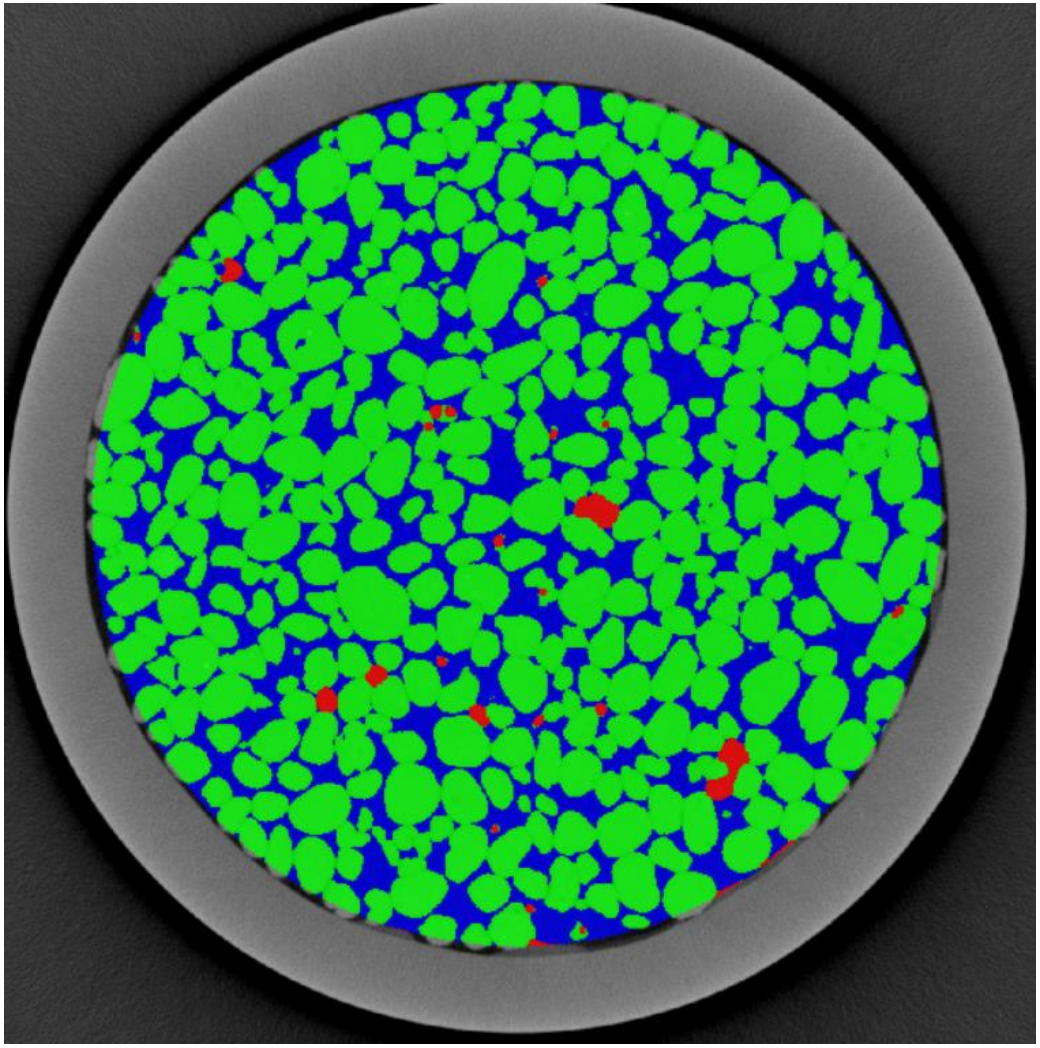
  - Eau + air
  - Sable
  - e

Classification 3D



Gestion des transitions

Classification 3D

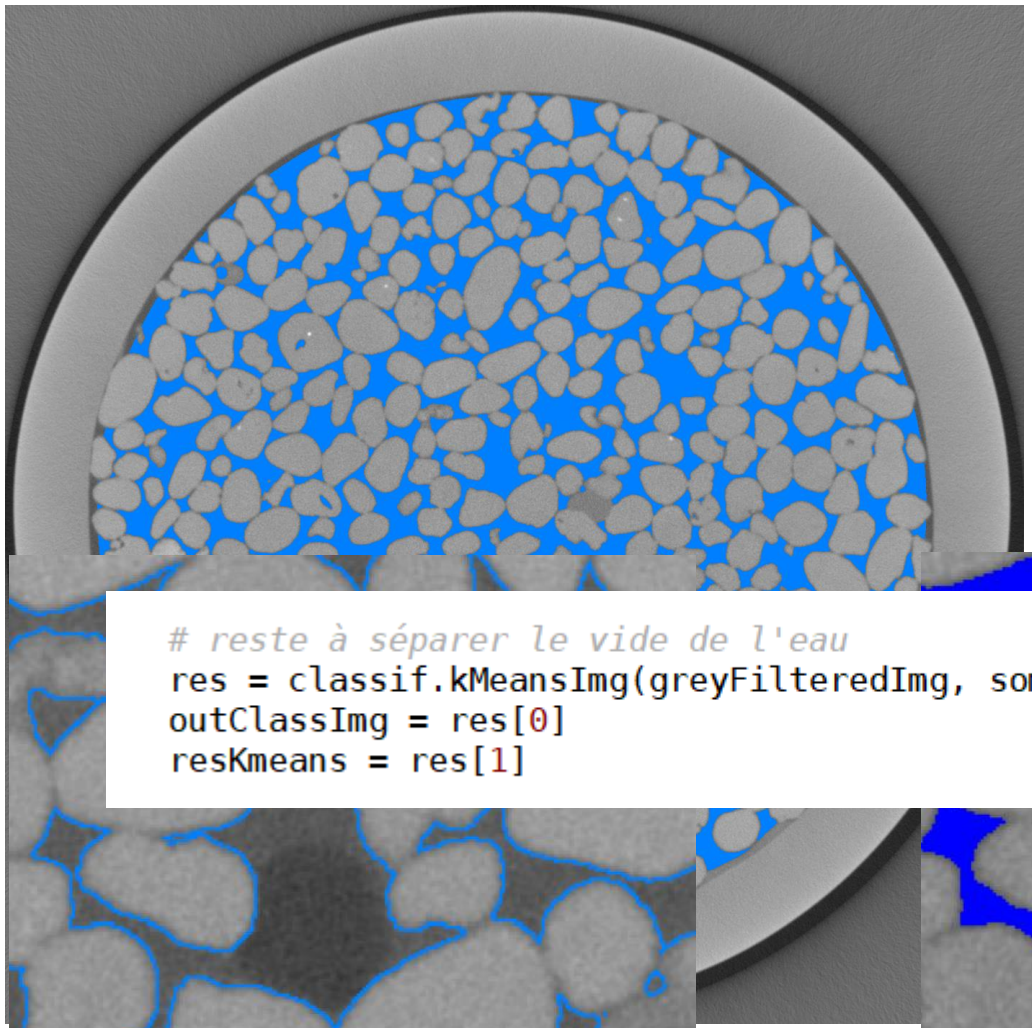


Première étape:

Classification en 3 classes

- Eau + air
- Sable
- Huile

Classification 3D



Deuxième étape:

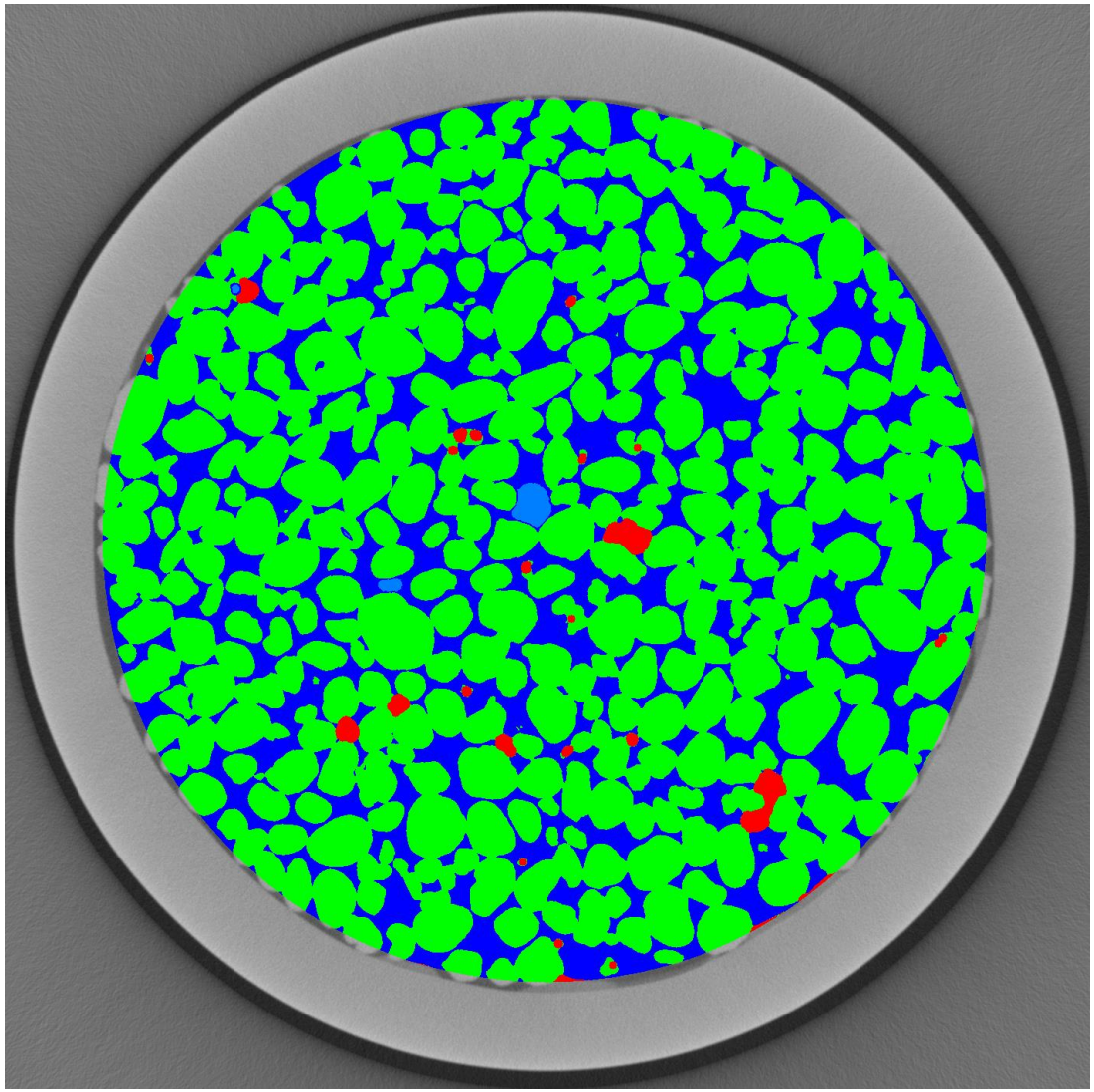
Classification en 2 classes

- Eau
- Air

Mais uniquement dans le masque

```
# reste à séparer le vide de l'eau
res = classif.kMeansImg(greyFilteredImg, sombreImg, 2)
outClassImg = res[0]
resKmeans = res[1]
```

Classification 3D



Recombinaison finale:

Classification en 4 classes

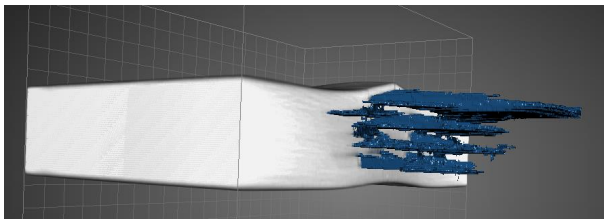
- Ratio surfacique
- Surface de contact
- Granulométrie

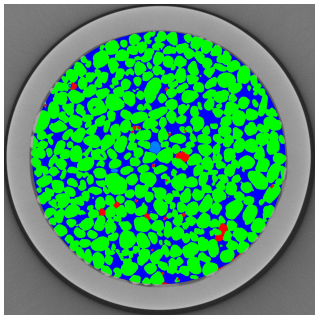
Conclusion

IPSDK peut vous permettre de traiter

- très finement vos images
- de façon extrêmement rapide

Reactiv'IP peut vous aider à mettre en œuvre ces solutions.





Merci pour votre attention

Questions ?

Reactiv'IP
163 cours Berriat
38000 Grenoble - France
Web: www.reactivip.com

Contact:
M. Laurent Bernard
Laurent.bernard@reactivip.com
Tél.: +33 (0)4 58 00 38 85