Scripts

Local Image Features Extraction
— LIFEx —

C. Nioche, F. Orlhac, I. Buvat

LIFEx version 7.5.n,
Last update of document: 2023/10/13

List of Figures

Part I
What is a script?

Chapter 1
Introduction

1.1 Rationale and what is a script?

Scripting is the process of writing scripts that automate certain tasks within LIFEx application. Scripting is often used in this application to automate repetitive tasks, perform complex calculations, or customize the user experience.

In other words, a script file makes it possible to run all operations and calculations without any user interaction. You can prepare it in advance. You can have as many script files as you want, with names that you choose and you can save them and modify them. They are simple text files.

The scripting procedure for LIFEx application typically involves the following steps:

- Writing the script: you can start writing the script itself. This involves using the syntax and commands of the scripting language to create a set of instructions that the application can execute.

- Debugging and testing the script: After writing the script, you will need to test it to make sure it works as intended. This involves running the script and checking its output, and debugging any errors or issues that may arise.
- Integrating/Executing the script with the application: Once the script is working correctly, you will need to execute it with the application. This involves specifying when and how the script should be executed within the application.
- Maintaining the script: Finally, you will need to maintain the script over time to ensure that it continues to work correctly as the application evolves and changes. This may involve updating the script to work with new versions of the application, fixing any bugs that arise, or adding new functionality as needed.

1.2 What are the existing scripts for LIFEx?

- General-Script: these scripts allow you to perform mass image format changes (format conversion). For example, if you want to export series from DICOM format to nifti format. The details of all these scripts are written in part II.
- Texture-Script: this script allows to automate a texture analysis on a whole cohort of patients. The details of all these scripts are written in part III.
- MTV-Script: this script allows to automate an MTV analysis on a whole cohort of patients. The details of all these scripts are written in part IV.

1.3 General information about writing a script text file

Introduction: A script file for LIFEx is a text file containing a sequence of properties. Each property is written on a line.

Properties: Properties are configuration values managed as key=value pairs. In each pair, the key and value are both String values. The key identifies and is used to retrieve, the value, much as a variable name is used to retrieve the variable's value.

Property example: For example, an application capable of saving file might use a property named "*LIFEx.Output.File*" to keep track of the directory used for the saving the file.

$LIFEx.Output.File = /home/user/results.csv$

- key property is "*LIFEx.Output.File*" Upper and lower case are not important in the key. This key can also be written : "*LIFEx.output.file*"
- value property is "*/home/user/results.csv*"

Construction guidelines of property:

- warning: a key is unique and cannot be repeated. Only one key per line.
- spaces before the key do not count.
- the order of the lines does not matter, however for human reading it is better to categorize the actions.
- file paths must always be written with a file separator "/" even under Windows.
- lines beginning with "#" are comments and are not interpreted by the application.

1.4 Script execution sequence

Here is the (mandatory) tasks order of execution in a script. Each script file will be read in full and executed as follows:

1. Series loading [mandatory]
2. — Series operations [optional]
3. — Series saving [optional]
4. — ROI loading [optional]
5. —— ROI operations [mandatory if "loaded ROI", otherwise optional]
6. —— ROI saving [mandatory if "loaded ROI", otherwise optional]
7. —— ROI extract features [optional and automatic]
8. —— ROI save result files [optional and automatic]
9. — ROI close [optional and automatic]
10. Series close [automatic]

1.5 How to run a script file

Before running a script:

1. Please check that you have no session file in the destination directory that you have specified in your script file;
2. The result filenames should NOT be `*TextureSession.csv`. This file name is already used in texture scripts.

Running the script :

- Please drag the script file in the panel used to load patient images.
- Alternatively, you can use the "browser + LocalDisk" interface to find and load your script file from your local disk.

1.6 Can I run several script files at the same time?

You can indeed run several script files one after another automatically. To do so, please select all script files and drag them in the panel used to put the patient image files. The script files will be executed one after another.

Beware: you can not load your script files using the "browser" interface, which supports only one script file at a time.

Part II
General concepts

Chapter 1
How to create a script?

1.1 Write a script

It is possible to chain the reading of the images and their respective ROI in order to obtain a including all results (.csv).

1.2 Minimal script

This section shows an script example. This script loads 2 series named PT0 and PT1, and then 2 ROI associated with PT0 (in Patient 0) and 2 ROI associated with PT1 (in Patient 1).

You can copy/paste this script into a text file named script.txt (for example). Don't forget to change some information: ex. directory/subDirectory

1.2 Minimal script

```
#______________________________________________________________________________
#
# Common
#______________________________________________________________________________

# result file -> mandatory
LIFEx.Output.File={directory/subDirectory}/results.csv

#______________________________________________________________________________
#
# Patient 0 / Series 0
#______________________________________________________________________________
# loading series -> mandatory
LIFEx.Patient0.Series0={directory/subDirectory}/PT0

#______________________________________________________________________________
#
# Patient 1 / Series 0
#______________________________________________________________________________
# loading series -> mandatory
LIFEx.Patient1.Series0={directory/subDirectory}/PT1

#______________________________________________________________________________
#
# Patient 0 / ROI 0
#______________________________________________________________________________
# loading ROI -> mandatory
LIFEx.Patient0.Roi0={directory/subDirectory}/RoiVolume/R1.uint16.nii.gz
LIFEx.Patient0.Roi1={directory/subDirectory}/RoiVolume/R2.uint16.nii.gz

#______________________________________________________________________________
#
# Patient 1 / ROI 1
#______________________________________________________________________________
# loading ROI -> mandatory
LIFEx.Patient1.Roi0={directory/subDirectory}/RoiVolume/R1.uint16.nii.gz
LIFEx.Patient1.Roi1={directory/subDirectory}/RoiVolume/R2.uint16.nii.gz
```

The minimal script (without comments) can therefore also be written this way:

```
LIFEx.Script = Main

    LIFEx.Output.File={directory/subDirectory}/results.csv
LIFEx.Patient0.Series0={directory/subDirectory}/PT0
LIFEx.Patient1.Series0={directory/subDirectory}/PT1
LIFEx.Patient0.Roi0={directory/subDirectory}/RoiVolume/R1.uint16.nii.gz
LIFEx.Patient0.Roi1={directory/subDirectory}/RoiVolume/R2.uint16.nii.gz
LIFEx.Patient1.Roi0={directory/subDirectory}/RoiVolume/R1.uint16.nii.gz
LIFEx.Patient1.Roi1={directory/subDirectory}/RoiVolume/R2.uint16.nii.gz
```

1.3 Complete script: save anonymous DICOM Series from DICOM Series

See below for a complete script example of save anonymous DICOM Series from DICOM Series:

```
#______________________________________________________________________________
#
# Common
#______________________________________________________________________________
# fixed information on script -> mandatory
LIFEx.Script = Main
LIFEx.Script.Version = 7.4.5

#______________________________________________________________________________
#
# Patient 0 / Series 0
#______________________________________________________________________________
# loading series -> mandatory
LIFEx.Patient0.Series0={directory/subDirectory}/PT0
LIFEx.Patient0.Series0.Operation0=Save anonymous series dcm
#______________________________________________________________________________
#
# Patient 1 / Series 0
#______________________________________________________________________________
# loading series -> mandatory
LIFEx.Patient1.Series0={directory/subDirectory}/PT1
LIFEx.Patient1.Series0.Operation0=Save anonymous series dcm
```

1.4 Main remarks relevant to the script writing

- What are the image formats that can be managed using LIFEx scripts?
  - image files can be in NIfTI-1 format (.nii or .nii.gz). In this case, the complete pathway to the patient image file should be given in the script;
  - to load DICOM images, you must define the root directory of these images (without the final filename). All files included in this directory will be loaded;

- What are the ROI formats that can be managed using LIFEx scripts?
  - the NIfTI-1 and RTStruct are supported in LIFEx scripts. In both cases, the file name should include the full path to the file. In the case of RTStuct, all ROI are loaded without exception.
- Syntax of all pathways of files in LIFEx scripts:
  - the syntax of pathway <u>must neither contain accents</u> and nor <u>space</u>.
  - the syntax of pathway must not necessarily conform to your system rules:
    - Windows: Unit:/Directory/File.extension or Unit:/Your Directory/ for series of DICOM images ; example C:/Home/Users1/File1
    - Linux: /Your Directory/Your File.extension
    - MacOs: /Your Directory/Your File.extension

Chapter 2

Perform operations

2.1 Introduction

Some intermediate operations are possible on the ROI. They will be performed after reading the Series and ROI but before extracting the features and recording the results:

Main syntax of one operation: The syntax property for one operation is:

```
{key}.Operation0=nameOp0,arg1,arg2,...,argn
```

with "nameOp0" is the title of the button of the ROI action to be performed, arg1 the first argument, arg2 the second argument, ...

Main syntax of many operations to the same Series or ROI: The syntax property for many operation is:

```
{key}.Operation0=nameOp0,arg1,arg2,...,argn|nameOp1,arg1,arg2,...,argn
```

with "nameOp0" is the first operation, "nameOp1" is the second operation.

On this line the order (from left to right) of the operations is respected during execution.

Order operations between lines: If you have more than one operation to perform (on a different series) you can describe the actions with an order given by the key_operationN:
{key}.Operation0=...
{key}.Operation1=...
{key}.Operation2=...

2.2 Series operations

Perform operations

2.2.1 Series operations

- **Syntax examples:**
  - Saving series with anonymous DICOM field:
    LIFEx.Patient0.Series0.Operation0=Save anonymous dcm
  - Change unit of series in SUVbm:
    LIFEx.Patient0.Series0.Operation0=SUVbw
  - 2x2x2 Resampling series then saving result series in nifti format:
    LIFEx.Patient0.Series0.Operation0=resampling,2,2,2|save nii uint16
  - Change output directory of series saving:
    LIFEx.Output.Directory = {directory/subDirectory/}

- **Available Series operations [file/edit menu]:**
  - Save anonymous series in DICOM format (only if loaded series is in DICOM format too):
    LIFEx.Patient0.Series0.Operation0=**Save anonymous dcm**
  - Save series in DICOM format (only if loaded series is in DICOM format too):
    LIFEx.Patient0.Series0.Operation0=**Save dcm**
  - Save series in nrrd format:
    LIFEx.Patient0.Series0.Operation0=**Save nrrd**
  - Save series in nifti format (float 32 bits):
    LIFEx.Patient0.Series0.Operation0=**Save nii float32**
  - Save series in nifti format (uint 16 bits):
    LIFEx.Patient0.Series0.Operation0=**Save nii uint16**
  - Save series in ECAT format: (.v):
    LIFEx.Patient0.Series0.Operation0=**Save ecat**
  - Save MIP 3D Plans: Coronal, Sagittal, Axial in nifti format (float32):
    LIFEx.Patient0.Series0.Operation0=**Save MIP 3D Plans**
  - Change output directory of Series saving:
    LIFEx.Patient0.Series0.Operation0.Output.Directory = {directory/subDirectory/}
  - Change common output directory of all Series saving:
    LIFEx.Output.Directory = {directory/subDirectory/}
  - Unit of Y axis of series, choose between: kBq/mL || SUVbw || SUVlbm || SUVibw || SUVbsa || Cpx/vx || Gy/vx || %/vx || .# class || .# k Pa || HU || T || mL/100g || mL/100g/min || sec || RC (.###) || min-1 || Proba || # || #.# || #.## || #.### || #.####
    LIFEx.Patient0.Series0.Operation0=**SUVbw**

- Apply a spatial resampling "SpatialResampling":
  LIFEx.Patient0.Series0.Operation0=**SpatialResampling,SigmaX,SigmaY,SigmaZ**
  Example with Sigma (x=3 mm, y=3mm, z=3mm):
  LIFEx.Patient0.Series0.Operation0=SpatialResampling,3,3,3
  - float value Sigma is the FWHM/2 of kernels size in milimeter.

- Apply a spatial "LaplacienOfGaussian" filter:
  LIFEx.Patient0.Series0.Operation0=**LaplacienOfGaussianFilter,SigmaX,SigmaY,SigmaZ, CalculationDimension,WholeBody,PaddingMethod**
  Example with Sigma (x=2 mm, y=2mm, z=2mm), 3d calculation dimension, wholebody enable and reflect padding method:
  LIFEx.Patient0.Series0.Operation0=LaplacienOfGaussianFilter,2,2,2,3d,true,reflect
  - float value Sigma is the FWHM/2 of kernels size in milimeter.
  - 3d calculation dimension is between [2d, 3d]
  - wholebody is always true in script
  - padding method is between [reflect, periodic, edge, zero]

- Apply a spatial "Gaussian":
  LIFEx.Patient0.Series0.Operation0=**GaussianFilter,SigmaX,SigmaY,SigmaZ, CalculationDimension,WholeBody,PaddingMethod**
  Example with Sigma (x=2 mm, y=2mm, z=2mm), 3d calculation dimension, wholebody enable and reflect padding method:
  LIFEx.Patient0.Series0.Operation0=GaussianFilter,2,2,2,3d,true,reflect
  - float value Sigma is the FWHM/2 of kernels size in milimeter.
  - 3d calculation dimension is between [2d, 3d]
  - wholebody is always true in script
  - padding method is between [reflect, periodic, edge, zero]

- Apply a spatial "Mean" filter:
  LIFEx.Patient0.Series0.Operation0=**MeanFilter,kernelDiameterSizeVx, CalculationDimension,WholeBody,PaddingMethod**
  Example with sigma (x=2 mm, y=2mm, z=2mm), 3d calculation dimension, wholebody enable and reflect padding method:
  LIFEx.Patient0.Series0.Operation0=MeanFilter,3,3d,true,reflect
  - float value sigma is the FWHM/2 of kernels size in milimeter.
  - integer value diameter kernel size between [3, 5, 7, 9, 11, 13, 15]
  - 3d calculation dimension is between [2d, 3d]
  - wholebody is always true in script
  - padding method is between [reflect, periodic, edge, zero]

- Apply a spatial "Wavelets" filter:
  LIFEx.Patient0.Series0.Operation0=**WaveletsFilter,TransformAlongZAxis,TransformAlongYXAxis, FamilyName,Order,Level**
  Example with TransformAlongZAxis enable, TransformAlongYXAxis enable, coiflet familyName, 1 order and 1 level
  LIFEx.Patient0.Series0.Operation0=WaveletsFilter,true,true,coiflets,1,1,reflect
  - transformAlongZAxis is between [true, false]
  - transformAlongYXAxis is always true
  - familyName is between [coiflets, biorthogonal, daubechies, haar, reverse biorthogonal, symlets]
  - daubechies order is between [1, ..., 38]
  - symlets order is between [2, ..., 20]
  - coiflets order is between [1, ..., 17]
  - reverse/biorthogonal order is between [11, 13, 15, 22, 24, 26, 28, 31, 33, 35, 37, 39, 44, 55, 68]

- haar order is equal [1]
- Level is equal [1]
- padding method is between [reflect, periodic, edge, zero]

- Apply a spatial "Laws" filter:
  LIFEx.Patient0.Series0.Operation0=**LawsFilter,KernelSize,WholeBody,PaddindMethod**
  Example with (l3, l3, l3 Kernel size, WholeBody enable, and reflect Paddind method
  LIFEx.Patient0.Series0.Operation0=LawsFilter,l3,l3,l3,true,reflect
  - kernel size is between [l3, l5, e3, e5, s3, s5, w5, r5]
    with l3: level (1, 2, 1)
    with l5: level (1, 4, 6, 4, 1)
    with e3: edge (-1, 0, 1)
    with e5: edge (-1, -2, 0 , 2, 1)
    with s3: spot (-1, 2, -1)
    with s5: spot (-1, 0, 2, 0, -1)
    with w5: wave (-1, 2, 0, -2, 1)
    with r5: ripple (1, -4, 6, -4, 1)
  - wholebody is always true in script
  - padding method is between [reflect, periodic, edge, zero]

2.3 ROI operations

2.3.1 ROI operations

- **Syntax examples:**
  - Below is an example of an "Absolute threshold" (n) with MinThreshold=2.5 and MaxThreshold=50:
    LIFEx.Patient0.Roi0.Operation0=n,2.5,50
  - Below is an other example of a "Absolute threshold" (n) with MinThreshold=2.5 and MaxThreshold=50 followed by the saving of result ROI in nifti (.nii) format:
    LIFEx.Patient0.Roi0.Operation0=n,2.5,50|Save nii

- **Available ROI operations [threshold menu]:**
  - Absolute threshold:
    LIFEx.Patient0.Roi0.Operation0=**n,ValueOfMinThreshold,ValueOfMaxThreshold**
  - Percent threshold:
    LIFEx.Patient0.Roi0.Operation0=**n%,ValueOfPercentThreshold**
  - 40 Percent threshold:
    LIFEx.Patient0.Roi0.Operation0=**40%**
  - 70 Percent threshold:
    LIFEx.Patient0.Roi0.Operation0=**70%**
  - Peak:
    LIFEx.Patient0.Roi0.Operation0=**Peak**
  - Nestle threshold:
    LIFEx.Patient0.Roi0.Operation0=**Nestle,ValueOfThreshold**
  - PERCIST threshold (need to have previously loaded a ROI named Liver):
    LIFEx.Patient0.Roi0.Operation0=**PERCIST**

- **Available ROI operations [file/edit menu]:**
  - Selected all ROI
    LIFEx.Patient0.Roi0.Operation0=**SelectedAllRoi**
  - Save ROI in nifti (.nii) format file:
    LIFEx.Patient0.Roi0.Operation0=**Save nii**
  - Save ROI in DICOM (.dcm) format file:
    LIFEx.Patient0.Roi0.Operation0=**Save dcm**
  - Save ROI in comma separator value (.csv) format file:
    LIFEx.Patient0.Roi0.Operation0=**Save csv**
  - Save ROI in turkupetcentre (.dft) format file:
    LIFEx.Patient0.Roi0.Operation0=**Save dft**
  - Not yet implemented (request of Recovery Coefficient not available in script):
    LIFEx.Patient0.Roi0.Operation0=**Save dftrc**
  - Save ROI in Pmod (.nrdd) format file:
    LIFEx.Patient0.Roi0.Operation0=**Save nrrd**
  - Change output directory of ROI saving:
    LIFEx.Patient0.Roi0.Operation0.**Output.Directory** = {directory/subDirectory/}
  - Change common output directory of all ROI saving:
    LIFEx.**Output.Directory** = {directory/subDirectory/}

Perform operations

Part III
Texture-Script

Chapter 1
How to create?

Scripting procedure for texture calculation without user interaction

1.1 Rationale

When you have a large number of ROI and/or a large number of patients for which you want to calculate usual indices (SUV, Volume, ...) and/or histogram-based or textural features, it can be convenient to run a script that will do all the calculations for you without any user interaction. ROI have to be prepared beforehand. You can then run the script many times if you want to perform the calculations using different sets of parameters.

For instance: You have 10 ROI per patient and a set of 100 patients to be processed. You are interested in studying the impact of the *nbGreyLevels* parameter on the 10*100 ROI, by setting this parameter to 64, 128 and 256. You can create three script files that will all be identical except for the line that sets the *nbGreyLevels* parameter. Running the script will calculate all indices for you automatically and store them in csv files. You will be able to run all three scripts using a single command line, see the "Can I run several script files at once?" p.12.

Another example: You have 10 ROI per patient and a set of 100 patients to be processed. You are interested in studying the impact of the *spatial resampling* parameter on the 10*100 ROI, by setting this parameter to 2x2x2 mm and to 4x4x4 mm. You can write and run a first script by setting the *spatial resampling* parameter to 2x2x2 mm. Then duplicate the script and change 2x2x2 mm by

4x4x4 mm to produce a second script to be run. Running the two scripts will calculate all indices for you automatically and store them in csv files. Setting the spatial resampling parameter to 2x2x2 mm in the script file can be done as follows:

```
# SpatialResampling of Series0 of Patient0
LIFEx.Patient0.Series0.SpatialResampling.Spacing.z=2
LIFEx.Patient0.Series0.SpatialResampling.Spacing.y=2
LIFEx.Patient0.Series0.SpatialResampling.Spacing.x=2
```

Using scripts may produce numerous ROIs. To check that the resulting ROI are compatible with texture index calculation and before running such calculations that can be time-consuming, you can use the CheckTex button after the ROI creation and before proceeding with textural index calculation. This will check whether the ROI includes a single cluster and contain a number of voxels greater than that required for consistent textural feature calculation. If one of these two conditions is not met, a warning message will be displayed.

1.2 Can LIFEx itself generate scripts?

Yes, and it's a great way to get started. LIFEx is able to produce a script (texture only) from the interface. Before (or after) pressing the "Run" button to execute the texture, you can request the construction of the corresponding script by pressing the script button.

How to create?

There are 4 levels of automatic script generation:

- Script "extended + common section": produces a script in which the parameters that can be changed affect all the series and ROI of the script
- Script "extended + individual section": produces a script in which the parameters that can be changed affect only a selected series or ROI.
- Script "minimal + common section": is an "extended + common section" in which all comments are deleted.
- Script "minimal + individual section":is an "extended + individual section" in which all comments are deleted.

Theses scripts consists of several different sections:

- Introduction
- Common section
- Common Spatial resampling section
- Common Spatial filtering section
- Common Texture section
- Patient / Series / Roi section

All these sections contain the series paths, ROIs and parameters needed to redo the texture analysis you've just designed in the user interface. You can then drag this file into the application's series/images/scripts reading area to run the whole procedure. You can also use this file as a starting point for modifying certain parameters in order to re-execute a new analysis under the same conditions.

1.3 How to write a script file?

The script file describes a list of functions to be executed within LIFEx.

- a) Reading the patient (series) file;

- b) Perform spatial resampling on series
- c) Perform spatial filter on series
- d) Reading one or several ROI files for that patient;
- e) Perform spatial resampling on ROI(s) (idem of series)
- f) Setting the parameters involved in the index calculation;
- g) Calculation of the different indices;
- h) Writing results in a new or existing csv file;
- i) Closing all files;
- j) Back to step a) if needed.

Here is an example of script file based on an example provided with LIFEx. You can save it as `TextureScript.txt`. The lines starting with # are not interpreted by LIFEx, they only include comments.

```
## This extended-common script file is automatically generated.
## Note: lines beginning with # are comments and are not taken into account by the application.
##
## This script consists of several different sections:
## - Common section,
## - Common Spatial resampling section,
## - Common Spatial filtering section,
## - Common Texture section,
## - Patient / Series / Roi section,
##
## All these sections contain the series paths, ROIs and parameters needed to redo the texture analysis you've just designed in the user interface.
## You can then drag this file into the application's series/images/scripts reading area to run the whole procedure.
## You can also use this file as a starting point for modifying certain parameters in order to re-execute a new analysis under the same conditions.


## ________________________________________________________________________________
##
## Common section
## ________________________________________________________________________________

## main: script type (main is the default value, even if the line doesn't exist) LIFEx>=7.5.0) [ Texture ]
LIFEx.Script=Texture

## main: KeepTheLargestCluster (true is the default value, even if the line doesn't exist) LIFEx>=7.4.5) [ true || false ]
LIFEx.KeepTheLargestCluster=true

## main: output file for results (LIFEx>=7.4.5) [ an absolute pathname ]
LIFEx.Output.File=/home/chris/data/PT_SUV.nii_0_0_1/Scripts/1697127304976_TextureResults.csv

## main: display roi messages on graphic user interface (false is the default value, even if the line doesn't exist) (LIFEx>=7.4.5) [ true || false ]
LIFEx.MessagesRoiGuiUpdate=false
## main: display roi and series on graphic user interface (true is the default value, even if the line doesn't exist) (LIFEx>=7.4.5) [ true || false ]
LIFEx.SeriesRoiGuiUpdate=true
```

```
## ____________________________________________________________________________________________
##
## Common Spatial Resampling section
##
## This section allows you to take Spatial Resampling parameters and make them common to the
analysis of all patients in the script.
##
## Remove comments ('#') to make it active.
## If you enable this common section, the equivalent section for each series is automatically disabled.
## ____________________________________________________________________________________________

## value information: (0 or 0.0 = no spatial resampling = native spacing voxels of selected series)

## SpatialResampling.Spacing.z (LIFEx>=5.1.0) [ unit mm ]
LIFEx.SpatialResampling.Spacing.z=0.0

## SpatialResampling.Spacing.y (LIFEx>=5.1.0) [ unit mm ]
LIFEx.SpatialResampling.Spacing.y=0.0

## SpatialResampling.Spacing.x (LIFEx>=5.1.0) [ unit mm ]
LIFEx.SpatialResampling.Spacing.x=0.0

## ____________________________________________________________________________________________
##
## Common Spatial Filtering section
##
## This section allows you to take Spatial Filtering parameters and make them common to the analysis
of all patients in the script.
##
## Remove comments ('#') to make it active.
## If you enable this common section, the equivalent section for each series is automatically disabled.
## ____________________________________________________________________________________________

## SpatialFiltering.LaplacianOfGaussian.Enable (LIFEx>=7.4.5) [ true || false ]
LIFEx.SpatialFiltering.LaplacianOfGaussian.Enable=true

## LaplacianOfGaussian.Sigma.z/y/x (LIFEx>=7.4.5) [ unit mm ]
LIFEx.SpatialFiltering.LaplacianOfGaussian.Sigma.z=1.000
LIFEx.SpatialFiltering.LaplacianOfGaussian.Sigma.y=1.000
LIFEx.SpatialFiltering.LaplacianOfGaussian.Sigma.x=1.000

## LaplacianOfGaussian.DimensionProcessing (LIFEx>=7.4.5) [ 3d || 2d ]
LIFEx.SpatialFiltering.LaplacianOfGaussian.DimensionProcessing=3d

## LaplacianOfGaussian.PaddingMethod (LIFEx>=7.4.5) [ WithoutPadding || WithPadding || Whole-
Body ]
LIFEx.SpatialFiltering.LaplacianOfGaussian.BoundingBox=WholeBody

## LaplacianOfGaussian.PaddingMethod (LIFEx>=7.4.5) [ reflect || periodic || edge || zero ]
LIFEx.SpatialFiltering.LaplacianOfGaussian.PaddingMethod=reflect

## ____________________________________________________________________________________________
##
## Common texture section
##
```

```
## This section allows you to take texture parameters and make them common to the analysis of all
patients in the script.
##
## Remove comments ('#') to make it active.
## If you enable this common section, the equivalent section for each series is automatically disabled.
## ______________________________________________________________________________________

## discretisation parameters: (LIFEx>=7.4.5)
## only one of the following two fields can be defined. These values depend on each other according
to the following equation:
## nbGreyLevels = ((maxBound - minBound) / binSize);
## If both values are set; only nbGreyLevels is taken into account for discretization
LIFEx.Texture.BinSize=0.0
LIFEx.Texture.NbGreyLevels=64.0

## discretisation function: Absolute (LIFEx>=5.1.0) [ true || false ]
LIFEx.Texture.Absolute=false
LIFEx.Texture.MinBound=0.0
LIFEx.Texture.MaxBound=20.0

## discretisation function: RelativeMeanSd (LIFEx>=5.1.0) [ true || false ]
LIFEx.Texture.RelativeMeanSd=false

## discretisation function: RelativeMinMax (LIFEx>=5.1.0) [ true || false ]
LIFEx.Texture.RelativeMinMax=true

## discretisation function: DistanceWithNeighbours (1 is the default value, even if the line doesn't
exist) (LIFEx>=5.1.0)
LIFEx.Texture.GLCM.DistanceWithNeighbours=01

## texture: dimension calculation (3d is the default value, even if the line doesn't exist) (LIFEx>=5.1.0)
[ 3d || 2d ]
LIFEx.Texture.DimensionProcessing=3d
## ______________________________________________________________________________________
##
## [Patient0] section
## ______________________________________________________________________________________
##
## [Patient0 / SeriesN] section
## ______________________________________________________________________________________

LIFEx.Patient0.Series0=/home/chris/loadV/LIFEx_protocolTexture_ControlQualitySet/ScriptDemoMultiSeries/PT_SUV.nii.gz

## ______________________________________________________________________________________
##
## [Patient0 / Roi0] section
## ______________________________________________________________________________________

LIFEx.Patient0.Roi0=/home/chris/loadV/LIFEx_protocolTexture_ControlQualitySet/ScriptDemoMultiSeries/R1.nii.gz


## ______________________________________________________________________________________
##
## [Patient0 / Roi1] section
## ______________________________________________________________________________________
```

LIFEx.Patient0.Roi1=/home/chris/loadV/LIFEx_protocolTexture_ControlQualitySet/ScriptDemoMultiSeries/R2.nii.gz

Here are the functions that are executed when running this text file:

- Reading the images (`PT_SUV.nii.gz`) of patient0
- No spatial resampling (because values=0)
- calculating of LaplacianOfGaussian spatial filtering (because enable=true) on series0
- Reading `R1.nii.gz` ROI of patient0
- Reading `R2.nii.gz` ROI of patient0
- Reading the setting of the Common, Absolute, RelativeMeanSd, RelativeMinMax
- Calculating all texture features of `R1.nii.gz` and `R2.nii.gz` on `PT_SUV.nii.gz`
- Writing the results in an existing csv session file named `1697127304976_TextureResults.csv`
- Closing all ROI and patient0 data

1.4 Scripting grey-level discretization

1.4.1 How set the quantization

How to create?

When you set the quantization of grey-level discretization parameters, you can leave one of the two sets to 0. Its value will then be automatically calculated based on the other parameter setting.

```
LIFEx.Texture.BinSize=3.125
LIFEx.Texture.NbGreyLevels=128.0
```

For instance:
if $BinSize = 0$ then $BinSize = (boundMax - boundMin)/(nbGreyLevels - 1)$
if $nbGreyLevels$ = 0 then $nbGreyLevels = ((boundMax - boundMin)/binSize) + 1$
if $BinSize$ = 0 and $nbGreyLevels = 0$ then $nbGreyLevels = 64$ by default
$boundMin$ and $bounsMax$ are the minimum and maximum values in the processed ROI.

Discretization has been set with:

$$discretizedValue = floor((nbGreyLevels * \frac{originalValue - boundMin}{boundMax - boundMin)} + 1)$$

1.4.2 Have the minimum and maximum bounds of a whole cohort of patients in a script (without texture calculation)

It is easy to find the BoundMax of all the ROI of all your patients in order to put this bound in the final script. This has to be done in 2 steps with exactly the same script (by changing one line). Here is the process.

- step 1: add this line in the script: "LIFEx.Check=true". It allows calculating the min, max bounds of all the ROI in a single pass of the script.
- step 2: run the script a first time to have only bound results; You then retrieve the max of the max of all the ROIs and correct the value in the script: LIFEx.Texture.MaxBound=found value
- step 3: change LIFEx.Check=false in place of LIFEx.Check=true in the script file
- step 4: run the script a second time to have all feature results

1.5 Scripting spatial resampling

When you set the spatial resampling parameters, you can leave the three parameters:
SpatialResampling.Spacing.z,
SpatialResampling.Spacing.y,
SpatialResampling.Spacing.x
set to 0 (or delete rows). This is equivalent to performing the calculation using the original voxel size of the images.

If you want to change the voxel size of the image, set the parameters to a value expressed in millimeters. The three mandatory values (*SpatialResampling.Spacing.z*, *SpatialResampling.Spacing.y*, *SpatialResampling.Spacing.x*) can be different.

1.6 Scripting main setting

LIFEx.Check: if true, allows to define a fast calculation of the script with only a few pre-selected features (voxelsCounting, min, mean, max). This key is used to produce a fast 1st pass on a patient coohort in order to establish the min and max of all ROIs of all patients. These values can be entered in the same script for a 2nd pass (with texture calculation : check=false).
LIFEx.check=false||true

LIFEx.KeepTheLargestCluster: allows you to define if you want to keep the largest of the clusters only when an ROI is composed of several clusters. If the value of this key is true, then only the largest cluster is kept (the others are deleted during the calculation). If this value is false, then the ROI remains composed of several clusters if this is the case.
LIFEx.keepOneLargest=false||true

LIFEx.Password: allows you to unlock a private function. Passwords are often used in our team to prepare new features that are not yet released. This allows the release of the application with hidden parts.
LIFEx.password=****

LIFEx.GuiUpdate: showing (true) or not showing (false) of graphical user interface (is java headless) (LIFEx>=7.4.1) [true || false]
This key allows to speed up the calculation of the scripts by not displaying the graphical user interface during the execution of the texture calculations (convenient for servers).
LIFEx.guiUpdate=false||true

LIFEx.SeriesRoiGuiUpdate: showing (true) or not showing (false) of series/ROI images in graphical user interface (LIFEx>=7.6.26) [true || false]
This key allows to speed up the calculation of the scripts by not displaying the different ROI and corresponding planar slices during the execution of the texture calculations.
LIFEx.seriesRoiGuiUpdate=false||true

LIFEx.MessagesGuiUpdate: showing (true) or not showing (false) of messages in graphical user interface (LIFEx>=7.6.26) [true || false]
This key allows to speed up the calculation of the scripts by not displaying the messages during the execution of the texture calculations.
LIFEx.messagesGuiUpdate=false||true

Part IV
MTV-Script

Chapter 1
How to create a MTV script

1.1 Introduction

The extraction of MTV values can be automated with scripts. It is possible to chain the reading of the images and their respective ROI in order to obtain a file including all results (.csv).

1.2 Script example

This section shows an script example of MTV protocol.

You can copy/paste this script into a text file named script.txt (for example). Don't forget to change some information: ex. directory/subDirectory

```
##########################################################################
# Common
##########################################################################
# fixed information on script -> mandatory
LIFEx.Script = MTV
LIFEx.Script.Version = 7.4.5

LIFEx.Check=false
LIFEx.KeepTheLargestCluster=false

# result file -> mandatory
LIFEx.Output.File={directory/subDirectory}/MTV_results.csv


##########################################################################
# Patient0 / Series
##########################################################################
# loading series -> mandatory
LIFEx.Patient0.Series0={directory/subDirectory}/PT0


##########################################################################
# Patient 1 / Series
##########################################################################
# loading series -> mandatory
LIFEx.Patient1.Series0={directory/subDirectory}/PT1


##########################################################################
# Patient 0 / ROI
##########################################################################
# loading ROI -> mandatory
LIFEx.Patient0.Roi0={directory/subDirectory}/RoiVolume/R1.uint16.nii.gz
LIFEx.Patient0.Roi1={directory/subDirectory}/RoiVolume/R2.uint16.nii.gz


##########################################################################
# Patient 1 / ROI
##########################################################################
# loading ROI -> mandatory
LIFEx.Patient1.Roi0={directory/subDirectory}/RoiVolume/R1.uint16.nii.gz
LIFEx.Patient1.Roi1={directory/subDirectory}/RoiVolume/R2.uint16.nii.gz
```

1.3 Output files

The root value of the key *"LIFEx.Output.File"* will be used for the construction of 2 files (_ROI and _SUMMARY) will be saved at the end of the script execution by the application:

- results_ROI_results.csv: will contain all the features extracted from each ROI;
- results_ROI_legend.csv: will contain the legend of all features extracted from each ROI
- results_SUMMARY_results.csv: will contain aggregated features calculated from several ROIs.
- results_SUMMARY_legend.csv: will contain the legend of aggregated features calculated from several ROIs.