

目录

简 介.....	2
第一章：医学图像 Python 库编程规范.....	3
1.1 标准的指导思想.....	3
1.2 代码布局方式及书写规范.....	3
1.2.1 制表符或空格.....	3
1.2.2 代码换行问题.....	3
1.2.3 空格在表达式中的运用.....	4
1.2.4 空行的使用.....	5
1.3 注释规范.....	5
1.4 命名规则.....	6
第二章：各种库的安装说明.....	8
2.1 所需函数库简介.....	8
2.2 在 Linux 系统中安装.....	8
2.2.1 Numpy.....	8
2.2.2 Pillow.....	8
2.2.3 Matplotlib.....	8
2.2.4 SimpleITK.....	9
2.2.5 VTK.....	9
2.3 在 Windows 系统中安装.....	9
2.3.1 Numpy.....	9
2.3.2 Pillow.....	10
2.3.3 Matplotlib.....	10
2.3.4 SimpleITK.....	11
2.3.5 VTK.....	12
2.4 pymesh 安装.....	12
2.5 小结.....	12
第三章：数据格式转换.....	13
3.1 文件格式介绍：.....	13
第四章：图像滤波.....	15
4.1 图像增强.....	15
4.2 边缘检测.....	17
4.3 图像平滑.....	19
4.4 边缘保留平滑.....	21
第五章 图像分割.....	24
5.1 简介.....	24
5.2 代码示例.....	24
第六章 图像配准.....	27
6.1 简介.....	27
6.2 二维配准.....	28
6.3 三维配准.....	32
第七章：曲面处理.....	36

简介

医学影像处理是目前产业界和科研领域的热门研究方向，全球每年有大量的科学研究和产业化项目围绕医学影像展开。近年来兴起的深度学习技术在医学影像智能诊断方面取得了迅速突破，引起了广泛的关注。

现今已经诞生了不少的编程语言，它们都有各自不同的特性：强大者如 Lisp，快速如 C，运用广泛如 Java，论古怪则如 Haskell。与这些语言不同，Python 将语言的很多特性进行了融合，将精力集中于引导开发者编写易读的代码，使用 Python 开发可谓轻而举。此外，Python 的易读性和简明性使它已经成为全球最流行的科研编程语言，新兴的深度学习技术也主要基于 Python 编程，网上有大量的 Python 函数库可供调用，Python 已成为科研编程的语言标准。

但是，仅针对医学影像分析研究的代码工具还是相对缺乏，本函数库是我们项目组构建的一个基础的医学图像处理的 python 函数库，放到著名的开源程序网站 GitHub 上向全球开源，欢迎同行下载使用，欢迎批评指正，提出宝贵意见。

研究组？

第一章：医学图像 Python 库编程规范

1.1 标准的指导思想

优美胜于丑陋：

python 以编写优美的代码为目标，注意编写的美观。

明了胜于晦涩：

优美的代码应当是明了的，命名规范，风格相似，不要为了追求技巧而使语言晦涩难懂。

简洁胜于复杂：

优美的代码应当是简洁的，不要有复杂的内部实现，将复杂的功能分解为简单的模块。

复杂胜于凌乱：

如果复杂不可避免，那代码间也不能有难懂的关系，要保持接口简洁。

扁平胜于嵌套：

优美的代码应当是扁平的，不能有太多的嵌套。

间隔胜于紧凑：

优美的代码有适当的间隔，不要奢望一行代码解决问题。

可读性很重要：

优美的代码是可读的，易读性非常重要。

1.2 代码布局方式及书写规范

1.2.1 制表符或空格

我们建议选择空格作为最优先的缩进方式。

Python 3 不允许混合使用制表符和空格来缩进。Python 2 的代码中混合使用制表符和空格的缩进，应该转化为完全使用空格。

调用 python 命令行解释器时使用 -t 选项，可对代码中不合法的混合制表符和空格发出警告(warnings)。使用 -tt 时警告(warnings)将变成错误(errors)。

这些选项可被用来检测是否混用制表符和空格，我们推荐编程时使用上述选项。

1.2.2 代码换行问题

一般，一个命令即一句代码占用一行。在遇到代码超长的情况下为了代码的美观及方

便测试，会涉及到代码的换行问题。

Python 字符串换行的三种方式：

第一种：三个单引号

```
python print (" 我是一个程序员 我刚开始学习 python")
```

第二种：三个双引号

```
python print (""" 我是一个程序员 我刚开始学习 python""")
```

第三种：\结尾

```
python print ("我是一个程序员，\ 我刚开始学 python")
```

() {} [] 中不需要特别加换行符：

```
python test2 = ('csdn' 'cssdn')
```

if 语句条件块太长需要写成多行：

值得注意的是两个字符组成的关键字（例如 if），加上一个空格，加上开括号，为后面的多行条件创建了一个 4 个空格的缩进。这会给嵌入 if 内的缩进语句产生视觉冲突，因为它们也被缩进 4 个空格。

多行结构中的换行问题：

多行结构中的结束花括号/中括号/圆括号应该是最后一行的第一个非空白字符或者是最后一行的第一个字符。

```
python my_list = [
```

```
1, 2, 3,
```

```
4, 5, 6,
```

```
]
```

```
result = some_function_that_takes_arguments(
```

```
'a', 'b', 'c',
```

```
'd', 'e', 'f',
```

```
)
```

行的最长长度

我们建议每行最大行宽不超过 80 个字符。

1.2.3 空格在表达式中的运用

各种括号内不要加空格

Yes: spam(ham[1], {eggs: 2})

No: spam(ham[1], { eggs: 2 })

逗号、冒号、分号前不要加空格，但是他们后面可以加

Yes: if x == 4: print x, y; x, y = y, x

No: if x == 4 : print x , y ; x , y = y , x

函数的左括号前不要加空格

Yes: spam(1)

No: spam (1)

序列的左括号前不要加空格

Yes: dict['key'] = list[index]

No: dict ['key'] = list [index]

操作符左右各加一个空格，不要为了对齐增加空格

Yes: x = 1

No: x=1

函数默认参数使用的赋值符左右省略空格

Yes: def complex(real, imag=0.0):

No: def complex(real, imag = 0.0):

1.2.4 空行的使用

- 1、两行空行用于分割顶层函数和类的定义
- 2、单个空行用于分割类定义中的方法
- 3、函数内逻辑无关段落之间空一行

1.3 注释规范

代码整体功能说明

整体功能的描述一般放置在代码最开头的注释中，用于对代码整体实现的功能进行阐述，以便于他人了解。

注意描述内容不要过于长，过于繁琐，让读者知道你的代码的编写目的就好。至于代码中具体某部分实现的功能，可用块注释进行解释。

Python 代码的整体注释是以多行注释组成的，为区别于单行注释，建议使用三引号注释法注释。python `""" Script Name : Author : Created : Last Modified : Version : Modifications : Description : """`

代码的块注释

最需要写注释的是代码中那些技巧性的部分。

如果你在下一次代码审查的时候必须解释一下，那么你应该现在就给它写注释。

对于复杂的操作，应该在其操作开始前写上若干行注释。

We use a weighted dictionary search to find out where i is in the array. We extrapolate position based on the largest num in the array and the array size and then do binary search to get the exact number.

param para1: 第一个参数的意义

param para2: 第二个参数的意义

param para3: 第三个参数的意义

rtype:

return:

func:

另一方面，不要描述代码，只需要对代码的功能，特殊的命名等你认为需要解释的部分加以阐述，以便他人快速阅读及自己修改使用。

同时，块注释应当与函数或类对齐，以使代码简洁美观。

单行注释

我们建议对于不是一目了然的代码，在其行尾添加注释。 `python NULL_BALLOT = Ballot(-1, -1)`
`# sorts before all real ballots`

1.4 命名规则

名称的英文缩写

常用的缩写，如 XML、ID 等，在命名时也应只大写首字母，如 XmlParser。
命名中含有长单词，对某个单词进行缩写。这时应使用约定成俗的缩写方式。
例如：

function 缩写为 func

text 缩写为 txt

object 缩写为 obj

count 缩写为 cnt

number 缩写为 num

变量的命名

全局变量

使用大写字母，单词之间用 '_' 分割，比如

NUMBER

COLOR_WRITE

而如果这个变量是内部的，那么在变量名前加 '_'

COLORWRITE

所谓"内部(Internal)"表示仅模块内可用, 或者, 在类内是保护或私有的。

其他变量

包括局部变量,实例变量, 静态变量。

使用小写字母, 单词之间用_分割, 比如

`thisisa_var`

同样, 如果这个变量是内部的, 那么在变量名前加'_'

`thisisavar`

类的命名

单词首字母大写, 即大驼峰命名法

`AdStats`

`ConfigUtil`

如果这个类是内部的, 那么在类名前加'_'

`_ConfigUtil`

函数的命名

使用小写字母, 单词之间用_分割:

`get_name()`

`count_number()`

同样, 如果这个函数是内部的, 那么在函数名前加'_'

`getname()`

常用缩写名称附录

例如:

- `dst = destination`
- `src = source`
- (未完待续)

1.5 其他建议

关于 `main()` 函数的使用

即使是一个打算被用作脚本的文件, 也应该是可导入的. 并且简单的导入不应该导致这个脚本的主功能(main functionality)被执行, 这是一种副作用. 主功能应该放在一个 `main()` 函数

中. 你的代码应该在执行主程序前总是检查 `if __name__ == '__main__':`, 这样当模块被导入时

主程序就不会被执行. `python def main(): ... if __name__ == '__main__': main()`

第二章：各种库的安装说明

2.1 所需函数库简介

在学习使用本函数库之前，有一些准备工作需要完成，以保证本函数库的正确使用。

首先，该函数库是在 Python3 版本下编写出来的，为保证代码可以正确运行，请使用 Python3。

由于该函数库旨在帮助使用者更简便的实现医学影像的处理，而非创造新的处理方式，我们在编写时借鉴引用了一些现有的开源函数库，所以在使用之前你需要保证你的计算机中安装了这些函数库，包括：Numpy, Pillow, Matplotlib, SimpleITK, VTK, pymesh。

其中，Numpy, Pillow, Matplotlib 都是较为基础且常见的 Python 函数库，在接下来的安装介绍中将不会详细阐述，而 SimpleITK, VTK 这两个更专注于医学图像处理的函数库，以及 pymesh 这个应用于曲面处理的库将偏重讲解。

另外，鉴于 OS X 系统中安装各种库与 Linux 中的方法相似，将不再单独讲解。

2.2 在 Linux 系统中安装

2.2.1 Numpy

首先，对于 Linux 用户，pip 安装较为简便，安装包版本也较新，是较为简洁的安装方式。该操作只需要一行代码：

```
1. 1. sudo pip install numpy
```

其次，也可以在官网下载源码包进行安装：

```
1. wget https://sourceforge.net/projects/numpy/files/latest/download
```

解压源码包：

```
2. unzip numpy-1.9.0.zip
```

进入解压目录：

```
3. cd numpy-1.9.0
```

运行解压目录里的 setup.py 文件

```
4. python setup.py install
```

2.2.2 Pillow

该库也可直接使用 pip 安装：

```
1. sudo pip install pillow==版本号
```

或者可以尝试：

```
2. sudo pip install -I --no-cache-dir -v Pillow
```

2.2.3 Matplotlib

同样，pip 安装仍旧可行：

1. `sudo pip install matplotlib`

或者也可以尝试下载源码安装包进行安装，操作流程与 2.2.1 中 Numpy 的源码安装相似，源码下载网址：<https://sourceforge.net/projects/matplotlib/files/matplotlib/>。

较新版本的 Python 对 Matplotlib 的安装可能会要求一些依赖库，可参考以下代码解决：

2. `sudo apt-get install libfreetype6-dev g++`

2.2.4 SimpleITK

在 Linux 下安装 SimpleITK 十分简单，即使是官网上也推荐 pip 安装，代码如下：

1. `sudo pip install SimpleITK`

也可以使用一些软件进行环境搭建，例如 Anaconda 等，在此不多介绍。

2.2.5 VTK

简便的安装方法是用 pip 等进行非编译安装，代码如下：

1. `sudo pip install vtk`

或者：

2. `sudo apt-get install python-vtk`

上述两种方式将实现相同效果。

如果你的计算机中已经有较完善的 cmake 等的基础，VTK 的安装也可以使用 g++ 或 cmake 等基础软件进行编译安装，以 cmake 为例：

首先下载 VTK：<http://www.vtk.org/download/>；

将文件解压到你想要的路径：

1. `sudo unzip VTK-7.0.0.zip`

准备编译文件目录：

2. `mkdir VTK_BUILD`
3. `cd VTK_BUILD`
4. `ccmake ../VTK/VTK-7.0.0`

执行 make

5. `sudo make`

安装：

6. `sudo make install`

2.3 在 Windows 系统中安装

2.3.1 Numpy

在 Windows 系统中，使用 pip 安装也是非常简便的方法：

1. `pip install numpy`

*.whl 文件下载安装也行得通，不过相较于 pip 安装较为麻烦：

下载 Numpy：<https://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy>

NumPy, a fundamental package needed for scientific computing with Python.

Numpy+MKL is linked to the Intel® Math Kernel Library and includes required DLLs in the numpy.core directory.

[numpy-1.14.5+mkl-cp27-cp27m-win32.whl](#)
[numpy-1.14.5+mkl-cp27-cp27m-win_amd64.whl](#)
[numpy-1.14.5+mkl-cp34-cp34m-win32.whl](#)
[numpy-1.14.5+mkl-cp34-cp34m-win_amd64.whl](#)
[numpy-1.14.5+mkl-cp35-cp35m-win32.whl](#)
[numpy-1.14.5+mkl-cp35-cp35m-win_amd64.whl](#)
[numpy-1.14.5+mkl-cp36-cp36m-win32.whl](#)
[numpy-1.14.5+mkl-cp36-cp36m-win_amd64.whl](#)
[numpy-1.14.5+mkl-cp37-cp37m-win32.whl](#)
[numpy-1.14.5+mkl-cp37-cp37m-win_amd64.whl](#)
[numpy-1.14.5-pp260-pypy_41-win32.whl](#)
[numpy-1.14.5-pp360-pp360-win32.whl](#)
[numpy-1.15.1+mkl-cp27-cp27m-win32.whl](#)

选择合适版本下载后在命令行窗口使用 `cd` 命令打开下载目录，输入代码：

1. `pip install 文件名.whl`

命令行窗口在安装完成后会给出提示。

2.3.2 Pillow

pip 安装：

1. `pip install pillow`

或者和 Numpy 相同，下载*.whl 文件安装，下载网址：

<https://www.lfd.uci.edu/~gohlke/pythonlibs/#pillow>

Pillow, a replacement for PIL, the Python Image Library, which provides image processing functionality and supports many file formats. Use `from PIL import Image` instead of `import Image`.

[PIL-2.0+dummy-py2.py3-none-any.whl](#)
[Pillow-3.4.2-cp36-cp36m-win32.whl](#)
[Pillow-3.4.2-cp36-cp36m-win_amd64.whl](#)
[Pillow-5.2.0-cp27-cp27m-win32.whl](#)
[Pillow-5.2.0-cp27-cp27m-win_amd64.whl](#)
[Pillow-5.2.0-cp34-cp34m-win32.whl](#)
[Pillow-5.2.0-cp34-cp34m-win_amd64.whl](#)
[Pillow-5.2.0-cp35-cp35m-win32.whl](#)
[Pillow-5.2.0-cp35-cp35m-win_amd64.whl](#)
[Pillow-5.2.0-cp36-cp36m-win32.whl](#)
[Pillow-5.2.0-cp36-cp36m-win_amd64.whl](#)
[Pillow-5.2.0-cp37-cp37m-win32.whl](#)
[Pillow-5.2.0-cp37-cp37m-win_amd64.whl](#)
[Pillow-5.2.0-pp260-pypy_41-win32.whl](#)
[Pillow-5.2.0-pp360-pp360-win32.whl](#)

其余操作均与安装 Numpy 相同。

2.3.3 Matplotlib

pip 安装：

1. `pip install matplotlib`

和 Numpy 相同，Matplotlib 也可以下载*.whl 文件安装，下载网址：

<https://www.lfd.uci.edu/~gohlke/pythonlibs/#matplotlib>

Matplotlib, a 2D plotting library.

Requires numpy, dateutil, pytz, pyparsing, kiwisolver, cyclor, setuptools, and optionally pillow, pycairo, tornado, wxpython, pyside, pyc

[matplotlib-2.2.3-cp27-cp27m-win32.whl](#)
[matplotlib-2.2.3-cp27-cp27m-win_amd64.whl](#)
[matplotlib-2.2.3-cp34-cp34m-win32.whl](#)
[matplotlib-2.2.3-cp34-cp34m-win_amd64.whl](#)
[matplotlib-2.2.3-cp35-cp35m-win32.whl](#)
[matplotlib-2.2.3-cp35-cp35m-win_amd64.whl](#)
[matplotlib-2.2.3-cp36-cp36m-win32.whl](#)
[matplotlib-2.2.3-cp36-cp36m-win_amd64.whl](#)
[matplotlib-2.2.3-cp37-cp37m-win32.whl](#)
[matplotlib-2.2.3-cp37-cp37m-win_amd64.whl](#)
[matplotlib-2.2.3-pp360-pp360-win32.whl](#)
[matplotlib-2.2.3.chm](#)
[matplotlib-2.x-windows-link-libraries.zip](#)
[matplotlib-3.0.0rc2-cp35-cp35m-win32.whl](#)
[matplotlib-3.0.0rc2-cp35-cp35m-win_amd64.whl](#)
[matplotlib-3.0.0rc2-cp36-cp36m-win32.whl](#)
[matplotlib-3.0.0rc2-cp36-cp36m-win_amd64.whl](#)
[matplotlib-3.0.0rc2-cp37-cp37m-win32.whl](#)
[matplotlib-3.0.0rc2-cp37-cp37m-win_amd64.whl](#)
[matplotlib-3.0.0rc2-pp360-pp360-win32.whl](#)
[matplotlib_tests-2.2.3-py2.py3-none-any.whl](#)
[matplotlib_tests-3.0.0rc2-py2.py3-none-any.whl](#)

其余操作均与安装 Numpy 相同。

2.3.4 SimpleITK

pip 安装:

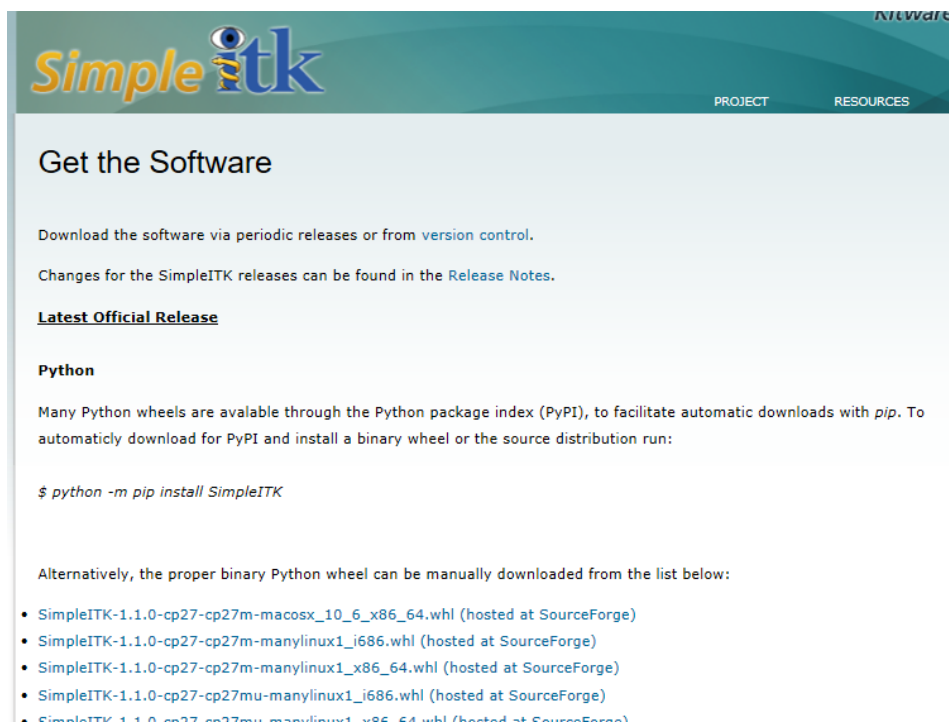
1. pip install SimpleITK

或:

1. python -m pip install SimpleITK

或者去官网下载*.whl 文件安装:

<http://www.simpleitk.org/SimpleITK/resources/software.html>



The screenshot shows the SimpleITK website's 'Get the Software' page. The header features the SimpleITK logo and navigation links for 'PROJECT' and 'RESOURCES'. The main content area is titled 'Get the Software' and includes instructions on how to download the software via periodic releases or from version control. It mentions that changes for SimpleITK releases can be found in the Release Notes. A section titled 'Latest Official Release' follows, with a sub-section for 'Python'. This section explains that many Python wheels are available through the Python package index (PyPI) and can be installed using pip. It provides a command to install SimpleITK: `$ python -m pip install SimpleITK`. Below this, it states that the proper binary Python wheel can be manually downloaded from a list of links. The list includes several wheels for different operating systems and architectures, all hosted at SourceForge.

之后与 Numpy 安装步骤相同。

2.3.5 VTK

pip 安装:

1. pip install vtk

*.whl 文件下载安装:

<https://www.lfd.uci.edu/~gohlke/pythonlibs/#vtk>

VTK (Visualization Toolkit), a software system for 3D computer graphics, image processing, and visualization.

VTK+qt4 requires pyqt4.

[VTK-5.10.1+qt486-cp27-none-win32.whl](#)

[VTK-5.10.1+qt486-cp27-none-win_amd64.whl](#)

[VTK-6.3.0-cp27-cp27m-win32.whl](#)

[VTK-6.3.0-cp27-cp27m-win_amd64.whl](#)

[VTK-7.1.1-cp27-cp27m-win32.whl](#)

[VTK-7.1.1-cp27-cp27m-win_amd64.whl](#)

[VTK-7.1.1-cp34-cp34m-win32.whl](#)

[VTK-7.1.1-cp34-cp34m-win_amd64.whl](#)

[VTK-7.1.1-cp35-cp35m-win32.whl](#)

[VTK-7.1.1-cp35-cp35m-win_amd64.whl](#)

[VTK-7.1.1-cp36-cp36m-win32.whl](#)

[VTK-7.1.1-cp36-cp36m-win_amd64.whl](#)

[VTK-8.1.1-cp35-cp35m-win32.whl](#)

[VTK-8.1.1-cp35-cp35m-win_amd64.whl](#)

[VTK-8.1.1-cp36-cp36m-win32.whl](#)

[VTK-8.1.1-cp36-cp36m-win_amd64.whl](#)

[VTK-8.1.1-cp37-cp37m-win32.whl](#)

[VTK-8.1.1-cp37-cp37m-win_amd64.whl](#)

然后进入下载目录，启动命令行窗口，输入命令:

1. pip install 文件名

完成安装。

2.4 pymesh 安装

Pymesh 的安装方法，其创作者已经在手册中有了较为详细的讲解，且暂时还没有其他特别优化的安装方法，在这里只给出安装介绍的网址:

<https://pymesh.readthedocs.io/en/latest/installation.html>

按照指导中的安装方法将能够完成安装。

2.5 小结

以上就是本医学图像处理的依赖库的安装方法，当然，还有很多不同的手段实现安装，就不全部一一列出了，以上所说的是较为普遍的手段。另外，各位用户在安装成功后，不要忘记在 python 中 import 各种库，来确定真的可以使用，以免造成使用本库函数时发生错误。

第三章：数据格式转换

医学图像因其特殊性，拥有与普通数字图像不同的文件格式。典型的医学图像格式包括 DICOM 图像格式、MHD 图像格式、Nii 图像格式、Nrrd 图像格式等。

本章实现调用 SimpleITK 库将 MDH 文件格式转化为 Nii 文件格式与 Nrrd 文件格式。

3.1 文件格式介绍：

MHD 文件格式

图形文件类型。 mhd 文件扩展名主要与图像元数据格式相关，图像元数据格式是医学图像的基于文本的标记文件格式。 它是 Insight Segmentation and Registration Toolkit 中使用的特殊图片格式，也是医学中使用的其他图形可视化软件的图片格式。

Nii 文件格式

nii 文件扩展名用于 NIfTI-1 数据格式，用于神经影像目的。 NIfTI-1 改编自广泛使用的 ANALYZE 7.5 数据文件格式。 此 nii 文件类型条目已标记为已过时且不再支持文件格式。

Nrrd 文件格式

nrrd 文件扩展名与 Nearly Raw Raster Data 文件格式相关联，支持科学可视化和涉及 N 维栅格数据的图像处理。

源代码：

```
1.  #!/usr/bin/env python
2.  # -*- coding: UTF-8 -*-
3.  """
4.  Script Name : ReadMhd_WriteOther
5.  Author      : ZengXiao
6.  Created     : 2018/4/5
7.  Modified    : 2018/4/7
8.  Version     : 2.0
9.  Description :
10. PURPOSE:
11.     Convert Mhd to nii and nrrd format.
12.
13. INPUTS:
14.     path: The path of the file which to be converted.
15.     Type of data: str
```

```

16. my_format: The format we want to convert to.
17.     Type of data: str
18.
19. RETURNED VALUE:
20.     The converted file named 'result' in the same path.
21.
22. Modified :
23.     Add notes and rewritten the codes as function.
24.
25. """
26. import SimpleITK as sitk
27.
28.
29. def format_conv(path, my_format):
30.     # Define function format_conv
31.     src = sitk.ReadImage(path)          # Call SimpleITK to read the image
32.     img_array = sitk.GetArrayFromImage(src) # Convert image to array
33.     img = sitk.GetImageFromArray(img_array) # Convert array to image
34.
35.     # Get the path the file located
36.     l = len(path)
37.     for i in range(0, l):
38.         if path[l - 1 - i] == '\\':      # Find the last \
39.             end = l - i
40.             break
41.
42.     self_path = path[0:end]              # Get the path of file
43.
44.     if my_format == 'nii':                # chose which format to save
45.         sitk.WriteImage(img, self_path + 'result.nii') # save the file named result in the same path
46.         print('format_conv')              # Tell the user function worked successfully
47.     elif my_format == 'nrrd':
48.         sitk.WriteImage(img, self_path + 'result.nrrd')
49.         print('format_conv')
50.     else:
51.         print('Sorry, We do not support this format.') # If there is an unexpected format, inform the error
52.
53.
54. """ Test function """
55. path = "D:\PythonCode\zx\src\PATIENT_DICOM.mhd" # Input path parameters
56. my_format1 = "nii" # Input format parameters
57. format_conv(path, my_format1) # Call function
58.
59. my_format2 = "nrrd" # Try another format

```

```
60. format_conv(path, my_format2)
61.
62. my_format3 = "txt"          # Try false format
63. format_conv(path, my_format3)
```

运行结果：

第四章：图像滤波

图像增强、边缘滤波、图像平滑、边缘保留平滑

4.1 图像增强

图像增强是为了使图像更适合于展示或者进一步研究而对数字图像进行处理的过程。常见的图像增强方式有噪点消除、图像锐化、对比度增强等，从而使得关键特征更容易被识别。

本节调用 SimpleITK 库进行图像增强，其方式有灰度扩张、自适应直方图均衡与直方图均衡。

灰度扩张

此前方式通过增加输入图像的灰度分布范围，增加原图像在低和高的灰度级上图像数据点，使图像的对比度增强。

自适应直方图均衡

该方法执行对比度受限的自适应直方图均衡。与直方图均衡不同，它在小数据区域而不是整个图像上运行。每个区域的对比度都得到增强，因此每个输出区域的直方图大致匹配指定的直方图（默认情况下均匀分布）。可以限制对比度增强，以避免放大可能存在于图像中的噪声。

拉普拉斯锐化

锐化滤波的作用是突出显示图像中的精细细节或增强模糊的细节。锐化是通过区分空间邻域差异完成的。

```
1. #!/usr/bin/env python
```

```

2.  #-*- coding: UTF-8 -*-
3.  """
4.  Script Name   : itkImageEnhancement
5.  Author        : ZengXiao
6.  Created       : 2018/4/20
7.  Version      : 1.0
8.  Description   :
9.  PURPOSE:
10.   Enhance images with the function of SimpleITK
11.
12. INPUTS:
13.   path: The path of the file which to be converted.
14.   Type of data: str
15.   my_method: The methods of Image Enhancement
16.   Type of data: str
17.
18. method:
19.   GSD: Gray scale Dilate
20.   AHE: Adaptive Histogram Equalization
21.   LS : Laplace Sharpening
22.
23. RETURNED VALUE:
24.   The enhanced image
25.
26. """
27. import SimpleITK as sitk
28.
29.
30. def itkImageEnhancement(path, my_method):
31.     # Define function itkImageEnhancement
32.     img = sitk.ReadImage(path)          # Call SimpleITK to read the image
33.     result = [];
34.
35.     # img = sitk.HistogramMatching(img1, img2) # Histogram Matching error: img can not be args
36.
37.     if my_method == 'GSD':              # chose the method of enhancement
38.         result = sitk.GrayscaleDilate(img) # save the file named result in the same path
39.         print('Gray Scale Dilate')       # Tell the user function worked successfully
40.     elif my_method == 'AHE':
41.         result = sitk.AdaptiveHistogramEqualization(img)
42.         print('Adaptive Histogram Equalization')
43.     elif my_method == 'LS':
44.         result = sitk.LaplacianSharpening(img)
45.         print('Laplace Sharpening')

```



```

46. else:
47.     print('Error') # Return an error
48.
49. return result
50.
51.
52. """ Test function """
53. path = "D:/PythonCode/zx/src/a.dcm" # Input path parameters
54. my_method = "AHE" # Method
55. image = itkImageEnhancement(path, my_method) # Call function
56.
57. my_method = "zx"
58. image = itkImageEnhancement(path, my_method) # Try incorrect method

```

结果：



Original



Gray scale dilate



Adaptive Histogram Equalization



Laplace Sharpening

4.2 边缘检测

边缘检测旨在使用各种各样的数学方法找到数字图像中灰度急剧改变或者不连续的点。

这些图像中灰度急剧变化的点的集合形成的曲线段被称为边缘。

高斯拉普拉斯边缘检测算子

通过卷积与高斯的一阶导数计算图像的梯度的大小。

有限差分法

使用简单的有限差分法计算每个像素的图像区域的梯度大小。

```
1. """
2.  Script Name  : itkImageEdgeFiltering
3.  Author      : Helin Yang
4.  Created     : 2018/4/19
5.  Version    : 3.0
6.  Description :
7.  PURPOSE    : Compute the edge of the image
8.  INPUTS     :
9.  - image_arr : The array of the image which to be converted.
10. - method    :
11. - Gass      : Computes the Magnitude of the Gradient of an image by convolution with the first derivativ
e of a Gaussian.
12. - Grad      : Computes the gradient magnitude of an image region at each pixel with a simple finite diffe
rence method.
13.  OUTPUT     : The edge of the image
14. """
15. import SimpleITK as sitk
16.
17. def itkImageEdgeFiltering(image_arr,method='Grad'):
18.
19.     if method == 'Gass':
20.         sitk.GradientMagnitudeRecursiveGaussian(image_arr)
21.         print('Gradient Magnitude Recursive Gaussian')
22.     elif method == 'Grad':
23.         sitk.GradientMagnitude(image_arr)
24.         print('Gradient Magnitude')
25.     else:
26.         print('Error')
27.
28.
29.
30. # An example
31. # Please don't forget to import those modules as follows
32. """
33. import itkImageEdgeFiltering
34. import SimpleITK as sitk
```

```

35. image = sitk.ReadImage('E:\IMAGE-BME\yanghelin\PATIENT_DICOM.mhd')
36. image_new = itkImageEdgeFiltering.itkImageEdgeFiltering (image,'Gass')
37. sitk.WriteImage( image_new ,\PATIENT_DICOM_new.mhd)
38. """

```

4.3 图像平滑

图像平滑通常用于降低图像噪声或产生较少像素的图像.图像平滑通常基于表示图像的单个值，例如图像的平均值或中间（中值）值。

高斯模糊

高斯模糊（也称为高斯平滑）是通过高斯函数模糊图像的结果。它是图形软件中广泛使用的效果，通常用于降低图像噪声并减少细节。

二项式模糊

二项式模糊通过多次重复基于帕斯卡三角形的 5x5 或 3x3 内核来模糊图像。如果方差大于 1，则结果非常接近真正的高斯模糊并且更快（对于低于~30 的方差）。

代码：

```

1. """
2.  Script Name  : itkImageSmoothing
3.  Author      : Boyce_Tian
4.  Created     : 2018/4/19
5.  Version    : 2.0
6.  Description :
7.  PURPOSE    : Serving several ways to smooth the image
8.  INPUTS     :
9.  - image_arr : The array of the image you want to smooth.
10. - type_str  :
11.   - DGauss  : Smooth image by function sitk.DiscreteGaussian()
12.   - Blur    : Smooth image by function sitk.BinomialBlur();
13.   - RGauss  : Smooth image by function sitk.RecursiveGaussian();
14.  OUTPUT      :
15.  - im_new   : An image as the result of smoothing.
16. """
17.
18. import SimpleITK as sitk
19.
20. def itkImageSmoothing(Im_arr , type_str):
21.
22.     # define the three ways

```

```

23. func_1 = 'Discrete Gaussian'
24. func_2 = 'Binomial Blurring'
25. func_3 = 'Recursive Gaussian'
26.
27. #get an image from the array input
28. image = sitk.GetImageFromArray(Im_arr)
29.
30. # find out the way to process the image according to type_str
31. # smooth the image
32. if type_str == func_1:
33.
34.     im_new = sitk.DiscreteGaussian(image)
35.
36. elif type_str == func_2:
37.
38.     im_new = sitk.BinomialBlur(image)
39.
40. elif type_str == func_3:
41.
42.     im_new = sitk.RecursiveGaussian(image)
43.
44. else:
45.     print('Please check your spelling,'
46.           'and try again.')
47.
48. return im_new
49.
50.
51. # an example of using the function
52. # smooth an image by the 'Discrete Gaussian' and save the output
53. if __name__ == '__main__':
54.     impath = './src_image/CT159.dcm'
55.     im= sitk.ReadImage(impath)
56.     image_arr = sitk.GetArrayFromImage(im)
57.     image_new=itkImageSmoothing(image_arr , 'Discrete Gaussian')
58.     sitk.WriteImage(image_new, './src_image/haha.dcm')

```

结果：



Original



Gaussian blur



Binomial blur

4.4 边缘保留平滑

边缘保持平滑是一种平滑纹理，同时保留锐利的边缘的图像处理技术。例如双边滤波，引导滤波和各向异性扩散滤波。当我们需要保存边缘信息，同时保留边缘，我们常使用边缘保留平滑。

双边滤波

双边滤波器是用于图像的非线性，边缘保持和降噪平滑滤波器。它用来自附近像素的强度值的加权平均值替换每个像素的强度。该权重可以基于高斯分布。至关重要的是，权重不仅取决于欧几里德像素距离，还取决于辐射度差异。因此可以保留锋利的边缘。

引导滤波

引导滤波源自局部线性模型，通过考虑引导图像的内容来计算滤波输出，引导图像可以是输入图像本身或另一不同图像。引导滤波器可以像流行的双边滤波器一样用作边缘保持平滑算子，但在边缘附近具有更好的行为。

各向异性扩散滤波

在图像处理和计算机视觉中，各向异性扩散（也称为 **Perona-Malik 扩散**）是一种旨在降低图像噪声而不去除图像内容的重要部分的技术，通常是对图像解释很重要的边缘，线条或其他细节。

源代码：

1. """

```

2.  Script Name : itkEdgePreservedSmoothing
3.  Author      : Han
4.  Created     : 2018/5/3
5.  Version     : 1.1
6.  Description :
7.  PURPOSE    : This is a function that smooth image with edge preserved
8.  INPUTS     :
9.  - im_arr    : Im_arr is the array of the image
10.             Type of data: ndarray
11. - type_str   : the type of edge-preserved smoothing that you want to process
12.             Type of data: str
13.             'B' = 'BilateralImageFilter'
14.             'MC' = 'MinMaxCurvatureFlowImageFilter'
15.             'CF' = 'CurvatureFlowImageFilter'
16.             'CAD' = 'CurvatureAnisotropicDiffusionImageFilter'
17.             'GAD' = 'GradientAnisotropicDiffusionImageFilter'
18.
19.  OUTPUTS    :
20.  - im_new     : Image
21.  """
22.
23.
24.  import SimpleITK as sitk
25.
26.
27.  def itkEdgePreservedSmoothing(im_arr, type_str):
28.
29.      func_1 = 'B'
30.      func_2 = 'MC'
31.      func_3 = 'CF'
32.      func_4 = 'CAD'
33.      func_5 = 'GAD'
34.
35.      # get an image from the array input
36.
37.      image = sitk.GetImageFromArray(im_arr)
38.      image = sitk.Cast(image, sitk.sitkFloat32)
39.
40.      # find out the way to process the image according to type_str
41.      # smooth the image
42.      if type_str == func_1:
43.
44.          im_new = sitk.Bilateral(image)
45.

```

```

46. elif type_str == func_2:
47.
48.     im_new = sitk.MinMaxCurvatureFlow(image)
49.
50. elif type_str == func_3:
51.
52.     im_new = sitk.CurvatureFlow(image)
53.
54. elif type_str == func_4:
55.
56.     im_new = sitk.CurvatureAnisotropicDiffusion(image)
57.
58. elif type_str == func_5:
59.
60.     im_new = sitk.GradientAnisotropicDiffusion(image)
61.
62. else:
63.     print('Please check your spelling,'
64.           'and try again.')
65.     return im_new
66.
67.
68. # an example of using the function
69. if __name__ == '__main__':
70.
71.     impath = './src_image/CT159.dcm'
72.     image = sitk.ReadImage(impath)
73.     print(image.GetDimension(), image.GetPixelID(), image.GetPixelIDValue(), image.GetSize())
74.     image_arr = sitk.GetArrayFromImage(image)
75.     image_new = itkEdgePreservedSmoothing(image_arr, 'GAD')
76.     print(image_new.GetDimension(), image_new.GetPixelID(), image_new.GetPixelIDValue(), image_
new.GetSize())
77.     image_test = sitk.Cast(image_new, sitk.sitkInt16)
78.     print(image_test.GetDimension(), image_test.GetPixelID(), image_test.GetPixelIDValue(), image_
t.GetSize())
79.     sitk.WriteImage(image_test, './src_image/yi.dcm')

```

结果：



Original



Bilateral Image Filter



Guided filter



Anisotropic diffusion

第五章 图像分割

5.1 简介

图像分割 (Segmentation) 指的是将数字图像细分为多个图像子区域 (像素的集合) 的过程。图像分割的目的是简化或改变图像的表达形式, 使得图像更容易理解和分析。对于医学图像, 分割区域通常对应特定的解剖结构, 如大脑、心脏和病灶等。医学图像分割也可以看作是从图像中提取出感兴趣的解剖结构。

医学图像分割在临床上常用于对骨骼的分割、对血管的分割和对内脏的分割等等。常用的方法包括 Region Growing, Watershed, Level Set 和 Fuzzy Connectedness 等方法。

5.2 代码示例

使用 Region Growing 进行医学图像分割的代码如下:

```
def itkRegionGrow(im_arr, type_str, seedlist, lower=0, upper=1):

    func_1 = 'CT'
    func_2 = 'CC'

    # get an image from the array input
    image = itkEdgePreservedSmoothing.itkEdgePreservedSmoothing(im_arr, 'CF')
    image = sitk.GetImageFromArray(im_arr)
    image = sitk.Cast(image, sitk.sitkFloat32)

    # find out the way to process the image according to type_str
    if type_str == func_1:
        im_new = sitk.ConnectedThreshold(image, seedlist, lower, upper)

    elif type_str == func_2:
        im_new = sitk.ConfidenceConnected(image, seedlist)

    else:
        print('Please check your spelling,'
              'and try again.')

    return im_new
```

这其中, func_1 和 func_2 分别为 Region Growing 的两种类型 ConnectedThreshold 和 ConfidenceConnected, 可以选用适当的类型进行图像分割。

使用 Fuzzy Connectedness 进行图像分割的代码如下：

```
def itkVectorFuzzyConnectednessImageFilter(path, seedList, iterations=4, multiplier=4.5,
radius=1, replaceValue=1):
    img = sitk.GetArrayFromImage(sitk.ReadImage(path))
    result = sitk.VectorConfidenceConnected(img, seedList, iterations, multiplier, radius,
replaceValue)
    return result
```

使用 Level Set 可以对 3 维医学图像（比如 CT 图）进行图像分割，代码如下：

```
def itkImageLevelSetSegmentation(Im_arr, type_str, seedlist=0):

    # define the six way
    func_1 = 'SD'
    func_2 = 'GAC'
    func_3 = 'TH'
    func_4 = 'FA'
    func_5 = 'FAB'
    func_6 = 'LP'

    # get an image from the array input
    image = sitk.GetImageFromArray(Im_arr)

    # find out the way to process the image according to type_str
    # smooth the image
    if type_str == func_1:
        image = sitk.Cast(image, sitk.sitkFloat32)
        image_pre = sitk.CannyEdgeDetection(image)
        im_new = sitk.ShapeDetectionLevelSet(image, image_pre)

    elif type_str == func_2:
        seg = itkRegionGrow.itkRegionGrow(Im_arr, 'CC', seedlist)
        init_ls = sitk.SignedMaurerDistanceMap(seg, insideIsPositive=True,
useImageSpacing=True)
        im_new = sitk.GeodesicActiveContourLevelSet(init_ls, sitk.Cast(image,
sitk.sitkFloat32))

    elif type_str == func_3:
        seg = itkRegionGrow.itkRegionGrow(Im_arr, 'CC', seedlist)
        stats = sitk.LabelStatisticsImageFilter()
        stats.Execute(image, seg)
        lower_threshold = stats.GetMean(1) - 1.5 * stats.GetSigma(1)
        upper_threshold = stats.GetMean(1) + 1.5 * stats.GetSigma(1)
        init_ls = sitk.SignedMaurerDistanceMap(seg, insideIsPositive=True,
useImageSpacing=True)
```

```

        im_new = sitk.ThresholdSegmentationLevelSet(init_ls, sitk.Cast(image, sitk.sitkFloat32),
lower_threshold, upper_threshold)

    elif type_str == func_4:
        im_new = sitk.FastMarching(image, seedlist)

    elif type_str == func_5:
        im_new = sitk.FastMarchingBase(image, seedlist)

    elif type_str == func_6:
        seg = itkRegionGrow.itkRegionGrow(Im_arr, 'CC', seedlist)
        init_ls = sitk.SignedMaurerDistanceMap(seg, insideIsPositive=True,
useImageSpacing=True)
        im_new = sitk.LaplacianSegmentationLevelSet(init_ls, sitk.Cast(image,
sitk.sitkFloat32))

    else:

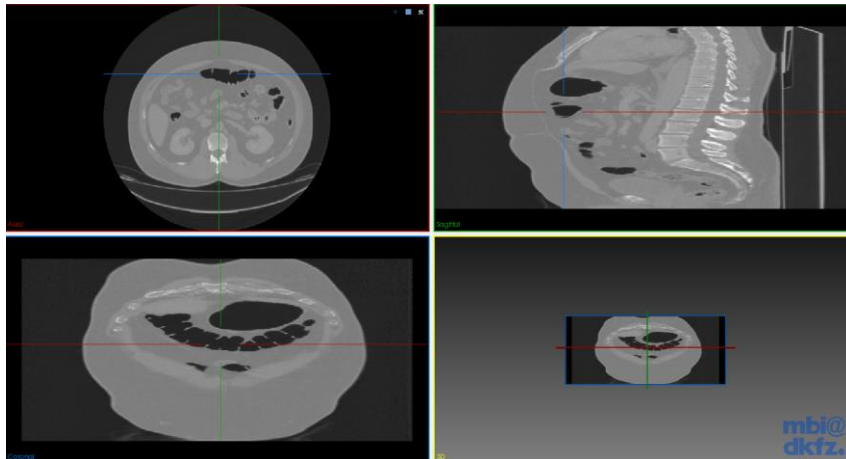
        print('Please check your spelling,'
              'and try again.')

    return im_new

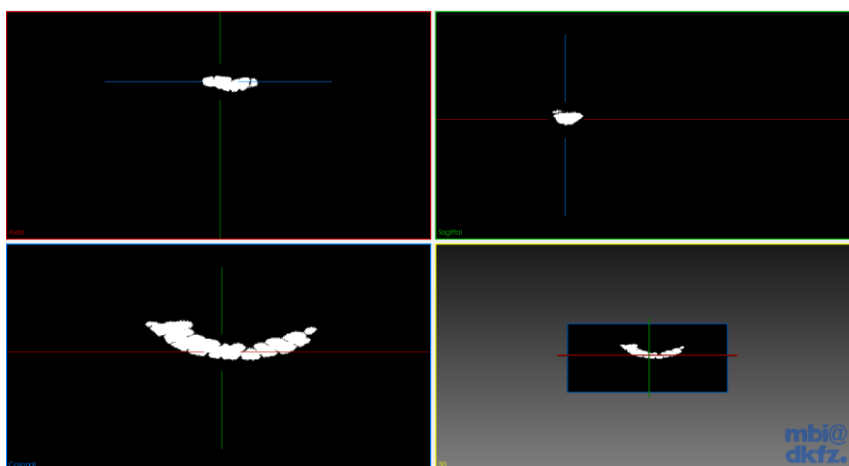
```

定义了六种类型的 Level Set。和前几种方法一样，输入为含图像信息的矩阵，输出也是包含图像信息的矩阵。

以下为原图：



以下为图像经过分割后的图：



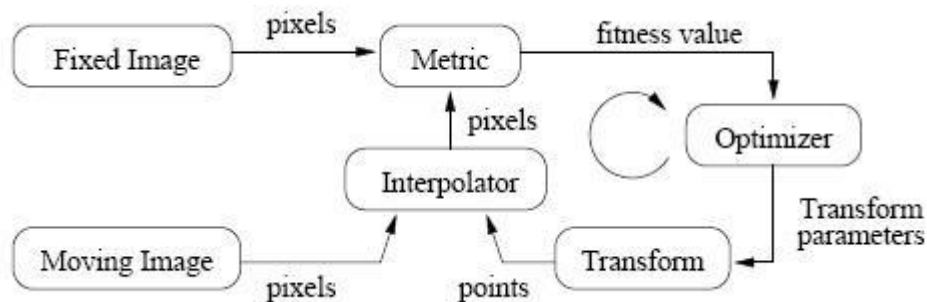
第六章 图像配准

6.1 简介

图像配准是将不同的数据集转换成一个坐标系的过程。数据可以是多张照片、来自不同传感器、时间、深度或视点的数据。图像配准或图像对齐算法可分为基于强度和基于特征的算法。其中一幅图像被称为移动图像或源图像，另一幅图像被称为目标图像、固定图像或感知图像。图像配准涉及到对源/移动图像进行空间变换，使其与目标图像对齐。目标图像的参考帧是静止的，而其他数据集被转换成与目标匹配的。

配准算法由以下四个要素构成的

- **Spatial Transform** （空间变换）
- **Similarity Metric** （相似性测度）
- **Optimizer** （优化方法）
- **Interpolator** （图像插值）



我们给出了基于 `simpleitk` 的一个函数，可以用以下参数来调用。

- **Spatial Transform** （空间变换）
 - 选择对移动图像施加何种变换，如以下种类

- 刚体变换 (rigid transform): 只包含平移和旋转, 可用于配制同一个人的头部 CT 和核磁图像
- 相似变换 (similarity transform): 包含平移、旋转和缩放
- 仿射变换 (affine transform): 包含平移、旋转、缩放和斜切、梯形变换, 可用于配准同一个人的胸部图像
- 非线性变换 (nonlinear transform): 包括仿射变换+扭曲, 典型的非线性变换包括 BSpline transform、optical flow 等。用于配准不同人图像
- **Similarity Metric** (相似性测度)
 - 衡量两个图像之间的相似度, 以便知道是否已经对齐。如以下种类:
 - 互相关 (Normalized Cross Correlation), 用于配准相同的影像模式, 如 CT 与 CT
 - 均方差 (Mean Squared Difference)
 - 互信息 (Mutual Information), 用于配准不同的影像模式, 如 CT 与 MR
- **Optimizer** (优化方法)
 - 是一种算法, 可以巧妙地找到最优的空间变换, 使得移动图像被变换后与固定图像的相似性测度最大。最优化方法有多种, 例如:
 - 共轭梯度法 (Conjugate Gradient Descendant), 稍复杂, 更有效
 - LBFGSB 方法
 - Powell 方法
 - 梯度下降法 (Gradient Descendant), 最简单方法
- **Interpolator** (图像插值)
 - 用于生成变换后的移动图像。可供选择的插值方法有:
 - 线性插值 (Linear Interpolation), 精度较好
 - 最邻近插值 (Nearest Neighbor Interpolation), 精度最低
 - B 样条插值 (B-Spline Interpolation), 精度最高

6.2 二维配准

```

1.  """
2.  Script Name  : itk2DImageRegistration
3.  Author      :
4.  Created     : 2018/5/31
5.  Version    : 1.0
6.  Description :
7.  PURPOSE    :
8.  INPUTS     :
9.  - Im_arr1   : fixed_image
10.             Type of data: not ndarray
11. - Im_arr2   : moving_image
12.             Type of data: not ndarray
13. - trans_type : Spatial Transform algorithms
14.             'Euler'   : 'Euler2DTransform'
15.             'BSpline' : 'BSplineTransformInitializer'
16.             'Similarity' : 'Similarity2DTransform'

```

```

17.         'Affine'      : 'AffineTransform'
18. - metric_type      : Similarity Metric algorithms
19.         'mutual'     : 'SetMetricAsMattesMutualInformation'
20.         'mean'       : 'SetMetricAsMeanSquares'
21.         'correlation' : 'SetMetricAsCorrelation'
22. - optimizer_type   : Optimizer algorithms
23.         'GradientLine': 'SetOptimizerAsConjugateGradientLineSearch'
24.         'LBFGSB'     : 'SetOptimizerAsLBFGSB'
25.         'Powell'     : 'SetOptimizerAsPowell'
26.         'Gradient'    : 'SetOptimizerAsGradientDescent'
27. - interpolator_type : Interpolator algorithms
28.         'Linear'     : 'sitkLinear'
29.         'Neighbor'   : 'sitkNearestNeighbor'
30.         'BSpline'    : 'sitkBSpline'
31. OUTPUTS :
32. - im_new : Image
33.
34. """
35.
36. import SimpleITK as sitk
37.
38. def itk2DImageRegistration(Im_arr1,Im_arr2,trans_type,metric_type,interpolator_type,optimizer_type):
39.     # define the transform way
40.     tfunc_1 = 'Euler'
41.     tfunc_2 = 'BSpline'
42.     tfunc_3 = 'Similarity'
43.     tfunc_4 = 'Affine'
44.
45.     mfunc1 = 'mutual'
46.     mfunc2 = 'mean'
47.     mfunc3 = 'correlation'
48.
49.     ifunc1 = 'Linear'
50.     ifunc2 = 'Neighbor'
51.     ifunc3 = 'BSpline'
52.
53.     ofunc1 = 'GradientLine'
54.     ofunc2 = 'LBFGSB'
55.     ofunc3 = 'Powell'
56.     ofunc4 = 'Gradient'
57.
58.
59.

```

```

60. # get an image from the array input
61. fixed_image = sitk.GetImageFromArray(Im_arr1,)
62. fixed_image = sitk.Cast(fixed_image, sitk.sitkFloat32)
63. moving_image = sitk.GetImageFromArray(Im_arr2)
64. moving_image = sitk.Cast(moving_image, sitk.sitkFloat32)
65.
66. # Spatial Transform algorithm depend on tfunc_ which are Euler,BSpline,Similarity,Affine
67. if trans_type == tfunc_1:
68.     transform = sitk.CenteredTransformInitializer(fixed_image,
69.                                                    moving_image,
70.                                                    sitk.Euler2DTransform(),
71.                                                    sitk.CenteredTransformInitializerFilter.GEOMETRY)
72.
73. elif trans_type == tfunc_2:
74.     transform = sitk.BSplineTransformInitializer(fixed_image, [fixed_image.GetDimension(), 8])
75.
76. elif trans_type == tfunc_3:
77.     transform = sitk.CenteredTransformInitializer(fixed_image,
78.                                                    moving_image,
79.                                                    sitk.Similarity2DTransform(),
80.                                                    sitk.CenteredTransformInitializerFilter.GEOMETRY)
81.
82. elif trans_type == tfunc_4:
83.     transform = sitk.CenteredTransformInitializer(fixed_image,
84.                                                    moving_image,
85.                                                    sitk.AffineTransform(fixed_image.GetDimension()),
86.                                                    sitk.CenteredTransformInitializerFilter.GEOMETRY)
87.
88. # Similarity Metric algorithms depend on mfunc which are mutual information, mean squares, correlation
89. registration_method = sitk.ImageRegistrationMethod()
90.
91. if metric_type == mfunc1 :
92.     registration_method.SetMetricAsMattesMutualInformation(numberOfHistogramBins=50)
93.
94. elif metric_type == mfunc2 :
95.     registration_method.SetMetricAsMeanSquares()
96.
97. elif metric_type == mfunc3 :
98.     registration_method.SetMetricAsCorrelation()
99.
100. registration_method.SetMetricSamplingStrategy(registration_method.RANDOM)
101. registration_method.SetMetricSamplingPercentage(0.01)
102.

```

```

103. #Interpolator algorithm depends on ifunc which are Linear, Nearest neighbor, BSpline
104. if interpolator_type == ifunc1:
105.     registration_method.SetInterpolator(sitk.sitkLinear)
106.
107. elif interpolator_type == ifunc2:
108.     registration_method.SetInterpolator(sitk.sitkNearestNeighbor)
109.
110. elif interpolator_type == ifunc3:
111.     registration_method.SetInterpolator(sitk.sitkBSpline)
112.
113. # Optimizer algorithms depends on ofunc which are Conjugate Gradient Line Search, LBFGSB, Powell, GradientDescent
114. if optimizer_type == ofunc1:
115.     registration_method.SetOptimizerAsConjugateGradientLineSearch(learningRate=1.0, numberOfIterations=70)
116.
117. elif optimizer_type == ofunc2:
118.     registration_method.SetOptimizerAsLBFGSB()
119.
120. elif optimizer_type == ofunc3:
121.     registration_method.SetOptimizerAsPowell()
122.
123. elif optimizer_type == ofunc4:
124.     registration_method.SetOptimizerAsGradientDescent(learningRate=1.0, numberOfIterations=60)
125.
126. registration_method.SetOptimizerScalesFromPhysicalShift()
127.
128. registration_method.SetInitialTransform(transform)
129. registration_method.Execute(fixed_image, moving_image)
130.
131. return transform
132.
133. """
134. When the.py file is run directly, the code block under if __name__ == '__main__' is run.
135. When the.py file is imported as a module, the code block under if __name__ == '__main__' is not run.
136. """
137. # if __name__ == '__main__':
138.
139. #input, preprocessing and registration
140.

```

调用函数示例：

```

1. path1 = 'E:/biomedicalIMAGE/MedImg_Py_Library-master/CT_Head_Patient1.mhd'
2. fixed_im = sitk.ReadImage(path1)

```

```

3. fixed_image_arr = sitk.GetArrayFromImage(fixed_im)
4. path2 = 'E:/biomedicalIMAGE/MedImg_Py_Library-master/MR_Head_Patient1.mhd'
5. moving_im = sitk.ReadImage(path2)
6. moving_image_arr = sitk.GetArrayFromImage(moving_im)
7. transform = itk2DImageRegistration( fixed_image_arr, moving_image_arr, 'BSpline', 'correlation', 'BSpline', 'GradientLine')

```

重新采样，将活动窗口固定到固定窗口

```

1. #resample and Rescale Intensity
2. resampler = sitk.ResampleImageFilter()
3. resampler.SetTransform(transform)
4. resampler.SetReferenceImage(fixed_im)
5. out = resampler.Execute(moving_im)
6. img0 = sitk.Cast(sitk.RescaleIntensity(moving_im), sitk.sitkUInt8)
7. img1 = sitk.Cast(sitk.RescaleIntensity(fixed_im), sitk.sitkUInt8)
8. img2 = sitk.Cast(sitk.RescaleIntensity(out), sitk.sitkUInt8)
9. img3 = sitk.Cast(sitk.RescaleIntensity(img1 / 2. + img2 / 2.), sitk.sitkUInt8)
10.
11. # combine these three scalar images into a multicomponent image named img_out
12. img_out = sitk.Compose(img1, img2, img3)
13. sitk.WriteImage(img_out, './src_image/re_Head_Patient1.mhd')

```

6.3 三维配准

```

1. #!/usr/bin/env python
2. # -*- coding: UTF-8 -*-
3. """
4. Script Name : itk3DImageRegistration
5. Author      : ZengXiao
6. Created     : 2018/5/29
7. Version     : 2.0
8. Description :
9. PURPOSE    :
10. INPUTS     : 3-D image registration
11. - fix      : Path of fixed image
12.             Type of data: str
13. - move     : Path of moving image
14.             Type of data: str
15. - transform : Spatial Transform

```



```

16.         Type of data: str
17.
18.         "euler": rigid transform
19.         "similarity": similarity transform
20.         "affine": affine transform
21.         "nonlinear": affine transform
22.
23.     - metric : Similarity Metric
24.         Type of data: str
25.
26.         "information": Mutual Information
27.         "ANTS": Unknown
28.         "correlation": Normalized Cross Correlation
29.         "Hinformation": Joint Histogram Mutual Information
30.         "MeanSquares": Mean Squared Difference
31.
32.     - interpolator: Optimizer
33.         Type of data: str
34.
35.         "linear": Linear Interpolation
36.         "nearest": Nearest Neighbor Interpolation
37.         "BSpline": B-Spline Interpolation
38.
39.     - optimizer : Substitution value
40.         Type of data: int
41.
42.         "AsGradient": Gradient Descendant
43.         "StepGradient": Conjugate Gradient Descendant
44.         "LBFGSB": LBFGSB
45.         "LBFGS2": LBFGS2
46.         "Exhaustive": Exhaustive
47.         "Amoeba": Amoeba
48.         "Weights": Weights
49.         "Powell": Powell
50.
51.     OUTPUTS : None
52.
53.     """
54.     import SimpleITK as sitk
55.     import os
56.
57.
58.     def itk3DImageRegistration(fix, move, transform, metric, interpolator, optimizer):
59.         switch_SetTransform = {

```

```

60.     "euler": sitk.Euler3DTransform(fixed_image.GetDimension()),
61.     "similarity": sitk.Similarity3DTransform(fixed_image.GetDimension()),
62.     "affine": sitk.AffineTransform(fixed_image.GetDimension()),
63.     "nonlinear": sitk.BSplineTransform(fixed_image.GetDimension()),
64. }
65. switch_metric = {
66.     "information": registration_method.SetMetricAsMattesMutualInformation(numberOfHistogramBins=50),
67.     "ANTS": registration_method.SetMetricAsANTSNighborhoodCorrelation(radius=50),
68.     "correlation": registration_method.SetMetricAsCorrelation(),
69.     "Hinformation": registration_method.SetMetricAsJointHistogramMutualInformation(numberOfHistogramBins=20,
70.                                             varianceForJointPDFSmoothing=1.5),
71.     "MeanSquares": registration_method.SetMetricAsMeanSquares(),
72. }
73. switch_interpolator = {
74.     "linear": sitk.sitkLinear,
75.     "nearest": sitk.sitkNearestNeighbor,
76.     "BSpline": sitk.sitkBSpline,
77. }
78. switch_optimizer = {
79.     "AsGradient": registration_method.SetOptimizerAsGradientDescent(learningRate=1.0, numberOfIterations=100,
80.                                                                     convergenceMinimumValue=1e-6,
81.                                                                     convergenceWindowSize=10),
82.     "StepGradient": registration_method.SetOptimizerAsRegularStepGradientDescent(relaxationFactor=0.5,
83.                                         gradientMagnitudeTolerance=1e-4,
84.                                         maximumStepSizeInPhysicalUnits=0.0),
85.     "LBFGSB": registration_method.SetOptimizerAsLBFGSB(gradientConvergenceTolerance=1e-5, numberOfIterations=500,
86.                                                         maximumNumberOfCorrections=5,
87.                                                         maximumNumberOfFunctionEvaluations=2000,
88.                                                         maximumStepSizeInPhysicalUnits=0.0),
89.     "LBFGS2": registration_method.SetOptimizerAsLBFGS2(numberOfIterations=0, hessianApproximateAccuracy=6,
90.                                                         deltaConvergenceDistance=0, deltaConvergenceTolerance=1e-5,
91.                                                         lineSearchMaximumEvaluations=40, lineSearchMinimumStep=1e-20,
92.                                                         lineSearchMaximumStep=1e20, lineSearchAccuracy=1e-4),
93.     "Exhaustive": registration_method.SetOptimizerAsExhaustive(stepLength=1.0),
94.     "Amoeba": registration_method.SetOptimizerAsAmoeba(parametersConvergenceTolerance=1e-8,
95.                                                         functionConvergenceTolerance=1e-4),
96.     "Weights": registration_method.SetOptimizerWeights(),

```

```

97.     "Powell": registration_method.SetOptimizerAsPowell(numberOfIterations=100, maximumLineIter
    ations=100,
98.                                     stepLength=1,
99.                                     stepTolerance=1e-6, valueTolerance=1e-6),
100. }
101. # read the images and casting the pixel type to Float32 (or Float64)
102. fixed_image = sitk.ReadImage(fix, sitk.sitkFloat32)
103. moving_image = sitk.ReadImage(move, sitk.sitkFloat32)
104.
105. # align the centers of the two volumes and set the center of rotation to the center of the fixed image.
106. initial_transform = sitk.CenteredTransformInitializer(fixed_image,
107.                                                       moving_image,
108.                                                       switch_SetTransform.get(transform, 'Incorrect parameter input'),
109.                                                       sitk.CenteredTransformInitializerFilter.GEOMETRY)
110. moving_resampled = sitk.Resample(moving_image, fixed_image, initial_transform,
111.                                   sitk.sitkLinear, 0.0, moving_image.GetPixelID())
112. registration_method = sitk.ImageRegistrationMethod()
113.
114. # Similarity metric settings.
115. switch_metric.get(metric, 'Incorrect parameter input'),
116. registration_method.SetMetricSamplingStrategy(registration_method.RANDOM)
117. registration_method.SetMetricSamplingPercentage(0.01)
118.
119. # Interpolator settings.
120. registration_method.SetInterpolator(switch_interpolator.get(interpolator, 'Incorrect parameter input'))
121.
122. # Optimizer settings.
123. switch_optimizer.get(optimizer, 'Incorrect parameter input')
124.
125. registration_method.SetOptimizerScalesFromPhysicalShift()
126.
127. registration_method.SetOptimizerScalesFromPhysicalShift(centralRegionRadius=5,
128.                                                         smallParameterVariation=0.01)
129. # Setup for the multi-resolution framework.
130. registration_method.SetShrinkFactorsPerLevel(shrinkFactors=[4, 2, 1])
131. registration_method.SetSmoothingSigmasPerLevel(smoothingSigmas=[2, 1, 0])
132. registration_method.SmoothingSigmasAreSpecifiedInPhysicalUnitsOn()
133.
134. # Don't optimize in-place, we would possibly like to run this cell multiple times.
135. registration_method.SetInitialTransform(initial_transform, inPlace=False)
136.
137. final_transform = registration_method.Execute(sitk.Cast(fixed_image, sitk.sitkFloat32),
138.                                               sitk.Cast(moving_image, sitk.sitkFloat32))
139.

```

```
140. # Query the registration method to see the metric value and the reason the optimization terminated.
141. print('Final metric value: {0}'.format(registration_method.GetMetricValue()))
142. print('Optimizer\'s stopping condition, {0}'.format(registration_method.GetOptimizerStopConditionDescription()))
143. sitk.WriteImage(moving_resampled, os.path.join('Output', 'D:\PythonCode\zx\src\
```

第七章：曲面处理

面绘制技术是利用计算机模拟光线照射到网格曲面上时，在屏幕上所投影得到的平面图像。现代计算机可以实现实时的一面绘制计算，即便当曲面发生旋转、平移、变形等运动时也可以实时地反应在屏幕上。

简单的一面绘制程序如下：

```
import vtk

filename = "Liver.stl"

reader = vtk.vtkSTLReader()
reader.SetFileName(filename)

mapper = vtk.vtkPolyDataMapper()

mapper.SetInputConnection(reader.GetOutputPort())

actor = vtk.vtkActor()
actor.SetMapper(mapper)

# Create a rendering window and renderer
ren = vtk.vtkRenderer()
renWin = vtk.vtkRenderWindow()
renWin.AddRenderer(ren)

# Create a renderwindowinteractor
iren = vtk.vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)

# Assign actor to the renderer
ren.AddActor(actor)

# Enable user interface interactor
iren.Initialize()
renWin.Render()
```

```
iren.Start()
```

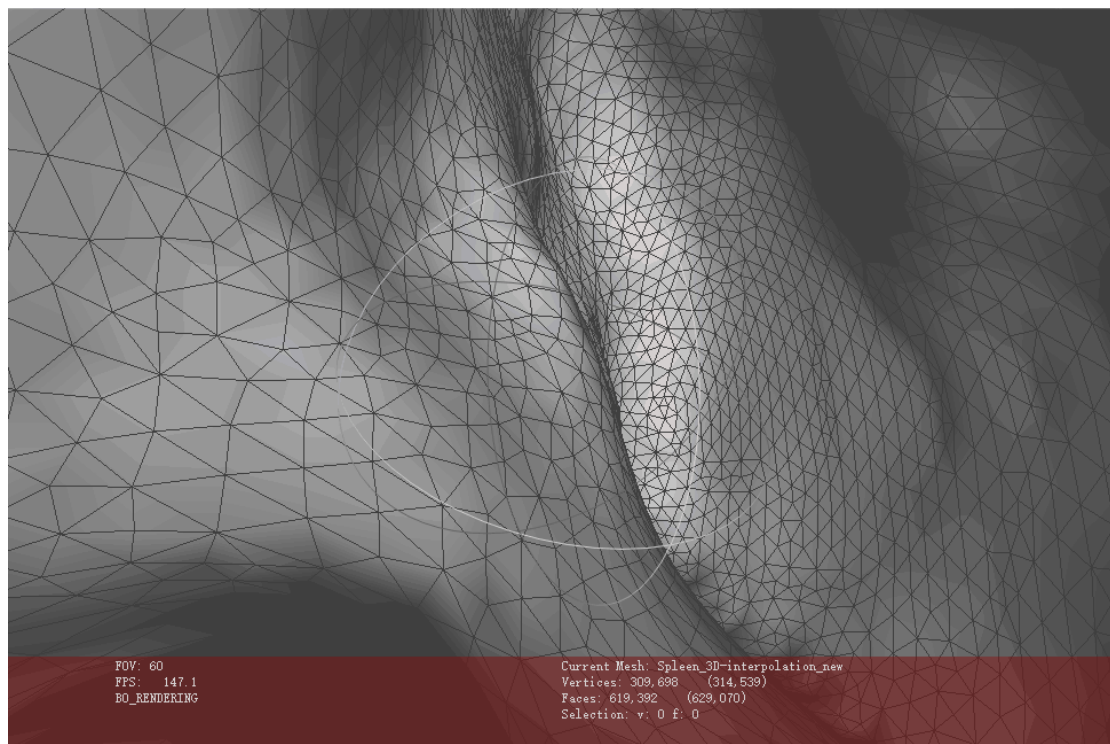
曲面处理的种类非常多，包括曲面平滑、网格简化、网格重构、曲面逻辑运算、细分曲面等等。

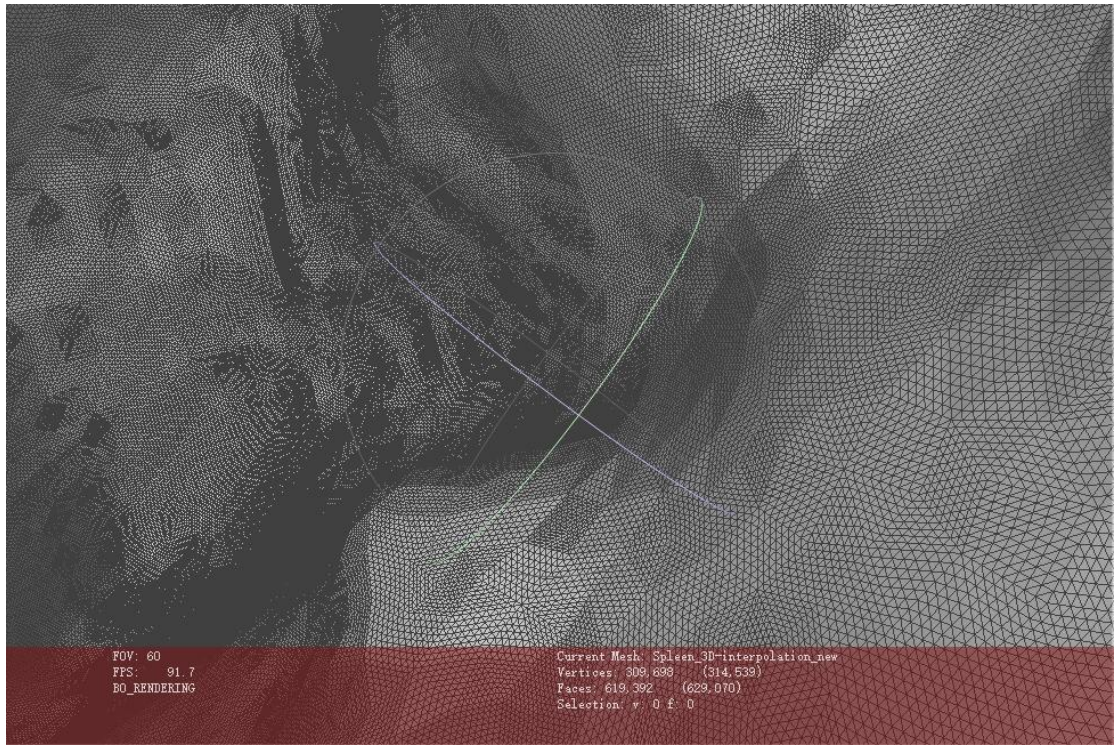
细分曲面是最简单的操作，代码如下：

```
import pymesh

# get the path of the original mesh and set the path of the Subdivided mesh
filepath="./Spleen_3D-interpolation.stl"
filenew="./Spleen_3D-interpolation_new.stl"
#get the original mesh
mesh=pymesh.load_mesh(filepath)
#subdivide the mesh
mesh_new=pymesh.subdivide(mesh,order=3,method='simple')
#write the subdivided mesh as a new file
pymesh.save_mesh(filenew,mesh_new)
```

效果如下：





FOV: 60
FPS: 91.7
BO_RENDERING

Current Mesh: Spleen_3D-interpolation_new
Vertices: 308,698 (314,538)
Faces: 619,392 (629,070)
Selection: v: 0 f: 0