基于SLIC的SuperVoxel分割算法

目录

[基于SLIC的SuperVoxel分割算法 1](#_Toc24934734)

[摘要 3](#_Toc24934735)

[一． SLIC算法分割SuperVoxel原理 4](#_Toc24934736)

[二． SLIC的Python实现 6](#_Toc24934737)

[2.1 python代码及其解释 6](#_Toc24934738)

[2.2 程序的运行结果 12](#_Toc24934739)

[三． 对SLIC分割SperVoxel的认识和思考 14](#_Toc24934740)

[1.1 参数m取值范围的变化 14](#_Toc24934741)

[1.2 灰度gray与亮度L 15](#_Toc24934742)

[1.3 压缩3D图像 16](#_Toc24934743)

[1.4 修改数据类型 17](#_Toc24934744)

[四． 参考文献 18](#_Toc24934745)

# 摘要

超像素概念是2003年Xiaofeng Ren提出和发展起来的图像分割技术，是指具有相似纹理、颜色、亮度等特征的相邻像素构成的有一定视觉意义的不规则像素块。它利用像素之间特征的相似性将像素分组,用少量的超像素代替大量的像素来表达图片特征,很大程度上降低了图像后处理的复杂度，所以通常作为分割算法的预处理步骤。已经广泛用于图像分割、姿势估计、目标跟踪、目标识别等计算机视觉应用。

目前常见的超像素分割算法大致可以分类为基于图或梯度上升的方法。基于图的算法有GS04, NC05, SL08, GCa10, GCb10b等，基于梯度上升的算法有W91, MS02, TP09, QS09, SLIC等。【1】

在之前的报告中介绍了SLIC(simple linear iterative clustering)，即简单线性迭代聚类算法。它在分割效果和分割速度两方面表现都比较优良，是一种比较实用的超像素分割算法。

但之前的报告只介绍了SLIC分割2D图像，而本文将在之前的基础上进行扩展，利用SLIC算法对3D医学图像进行SuperVoxel分割。本文将着重放在二维与三维情况的不同点进行描述。

本文首先介绍了SLIC算法在3D情况下的实现原理，关键是把二维的平面距离扩展为三维的空间距离，其余类似。

接下来本文通过python程序实现SILC,将展示每一部分python代码，并详细解释每一部分的功能。之后展示代码的实际运行效果。在安装二核的intel corei7-6500u处理器的笔记本电脑上分割大小为85\*61\*58像素的图片，迭代一次仅需要約18s，并通过IKT-SNAP软件【2】展示分割前后的3D图像，方便读者对比。

在本文的最后一部分将讲述作者本人对SILC进行SuperVoxel分割的认识和思考。

**关键词： SuperVoxel， SLIC，K-means聚类，python**

# SLIC算法分割SuperVoxel原理

若3D图片原由N个像素，要将其分割成K个SuperVoxel，那么平均每个SuperVoxel包含像素的个数是N/K。若SuperVoxel等大且在图片中均匀分布，呈正方体，那么每个SuperVoxel正方体的边长是



首先，将K个聚类中心均匀分布到图片的像素点上，每个聚类中心与其相邻的六个聚类中心的距离均为S。

接下来，将中心移动到与3\*3\*3邻域中的最低梯度位置相对应的种子位置。这样做是为了避免将聚类中心定位在边缘上，并且减少用噪声像素接种超像素的机会。

由于SuperVoxel的预期空间范围是近似尺寸S×S×S的区域，因此在超像素中心周围的区域2S×2S×2S中进行类似像素的搜索。每个像素可能会在8个聚类中心的搜索范围内，这里像素将与距离其最近的聚类中心相关联。

根据聚类算法的思想，我们需要构造一个距离函数D表示像素点到聚类中心的距离，在之前的报告中距离函数为



其中



表示当前像素的坐标，表示当前聚类中心的坐标。



表示当前像素在CIELAB空间中的值，表示当前聚类中心在CIELAB色彩空间中的值。

现在由于变为3D图像，空间距离函数修改为



并且由于由平面距离变为空间距离，Ds的取值范围变大，而Dc的取值范围与2维时相同，因此应该考虑减小常数m的取值范围，以保持Ds对距离函数D的影响所占比例不变。把m的最大值减小为最大值的2/3, 把m的取值范围由【1,40】减小为【1，25】。

按照上述描述，可以将所有的像素与最近的聚类中心相关联。然后聚类中心调整为属于该聚类中心的所有像素的平均值。然后在新的聚类中心周围的区域2S×2S×2S中进行类似像素的搜索。如此进行迭代。直到新的聚类中心的位置与先前的聚类中心的差值小于预期的阀值。

在本文并没有设置阀值来控制迭代的结束，而是通过人为的设置迭代次数来观察效果，最终确定一个合理的迭代次数。大多数时候迭代次数可以取10.【1】

经过上述迭代优化可能出现以下瑕疵：出现多连通情况、SuperVoxel尺寸过小，单个SuperVoxel被切割成多个不连续超像素等，这些情况可以通过增强连通性解决。可以将过小的SuperVoxel合并到距离其最近的SuperVoxel，以消除孤立的结点。

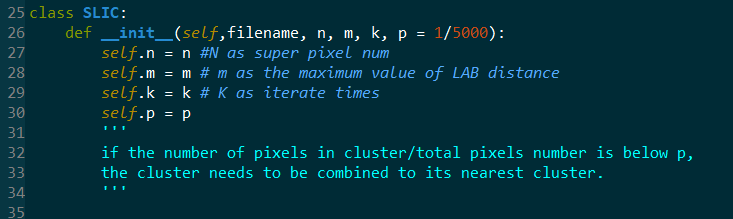
# SLIC的Python实现

## 2.1 python代码及其解释

在本文的python代码中，SLIC的实现过程主要是自定义了一个名为SLIC的类，通过各种内置方法完成SLIC算法中初始化聚类中心、移动聚类中心到3\*3\*3区域梯度最小的点，给像素分配聚类中心，生成SuperVoxel，迭代等过程。

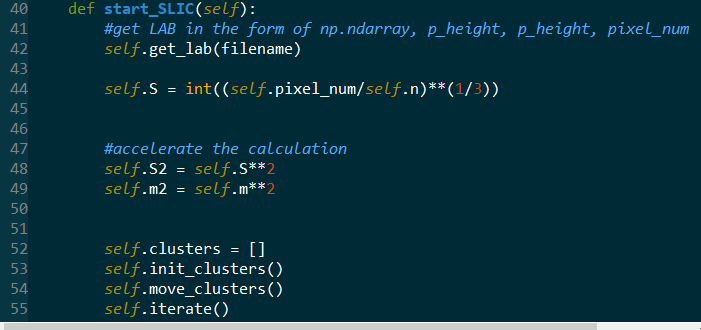
另有一点需要说明，本文使用的3D图像样例来源于ITK-SNAP软件的免费开源样例，格式为nii格式。

1. 定义SLIC类，并传入SLIC算法必须的参数。n代表要生成的SuperVoxel的个数，m代表上文提到的要归一化LAB距离所需要用到的LAB距离的最大值，k代表迭代次数。



1. 调用方法start\_SLIC，start\_SLIC方法是一个总控制，包含实现SLIC算法中一系列过程的各个方法。对start\_SLIC中调用的各个方法的解释见后文。

第44行代码计算了初始化聚类中心时两个相邻聚类中心的距离self.S,。第48,49行代码把self.S, self.m的平方计算出来并储存为类的属性，为的是避免在之后计算每个像素与聚类中心的距离时都需要重复计算这两个量，以缩短程序运行时间。52行的Self.clusters是一个存储所有的聚类中心的列表。



1. Get\_lab方法的作用是读入需要处理的image文件所在路径，通过调用专门处理医学图像的nibabel包,skimage包中的color.rgb2lab方法，把图片的RGB数据转换为LAB数据存储在self.lab中。并获取了图片像素的高度，高度，深度等信息。

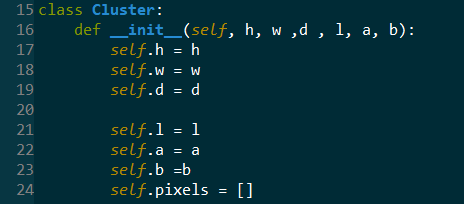
nii数据本身是除了 矩阵信息（也就是图像信息之外），还有和物理坐标相关的 仿射矩阵 以及 Header头文件 信息，但是nii的获取数据的方法get\_data()是一个单向操作，只可以获取数据，并不能把数据直接存到一个现成的nii文件中，因此需要记录之前图像的*仿射矩阵和头文件，*

（见61,62行），方便之后保存生成的nii图像时使用。

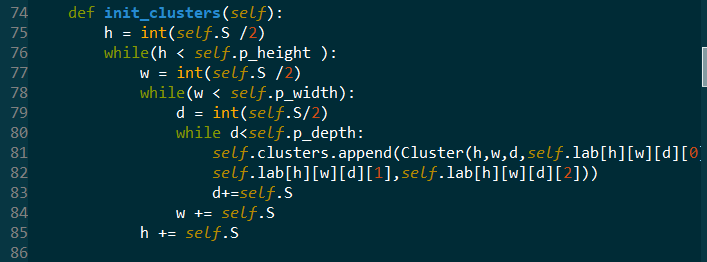
图片包含 文字

描述已自动生成

1. 为了方便存储聚类中心的信息，又定一个名为Cluster的类，其中的属性h,w,d,l,a,b分别代表一个聚类中心的像素坐标（h,w,d）和LAB空间中的值（l,a,b）。self.pixels代表属于此聚类中心的像素点的列表。



1. Init\_clusters的作用是初始化聚类中心。将聚类中心均匀分布在图片上，并将聚类中心cluster (type为（4）中定义的Cluster类)添加到self.clusters列表中。

\*3

1. Move\_clusters 在每个聚类中心3\*3\*3领域内选择出梯度最小的点作为新的聚类中心。这样做的目的是为了避免聚类中心落在梯度较大的轮廓边界，以影响后续聚类效果。代码96,98,100行的if语句是为了防止3\*3\*3领域超出图片像素范围。94和101行调用的self.get\_gradient（h,w,d）方法会返回坐标为（h,w,d）点的梯度值。

图片包含 文字

描述已自动生成

1. self.get\_gradient（h,w,d）方法会返回坐标为（h,w,d）点的梯度值。当像素点落在图片边沿（w=p\_width-1或h=p\_height-1时），返回的梯度值为无穷大inf。

图片包含 文字

描述已自动生成

1. iterate方法对给像素分配聚类中心，更新聚类中心的值，存储照片这三个过程进行迭代。

其中代码132行初始化距离矩阵distance, distance[i][j][l]代表坐标为[i,j,l]的像素点到与所有2S\*2S\*2S包含该像素点的聚类中心的距离的最小值。初始化时distance中所有的值都为无穷大inf。

Distance中元素的值的计算过程如下：当坐标为[I,j]的像素点在第一个聚类中心的2S\*2S\*2S领域时，distance[i][j]的值由inf更新为距离D1。此后，当第二个2S\*2S\*2S领域包含该像素的聚类中心出现时，计算距离D2,如果D2<D1,则把distance[i][j][l]的值更新为距离D2。当整个遍历过程结束，distance[i][j][l]存储着坐标为[I,j,l]的像素点到与所有2S\*2\*2S临域包含该像素点的聚类中心的距离的最小值。

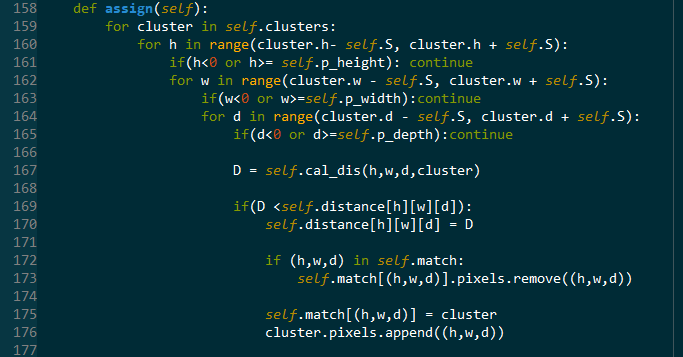
代码118行建立了一个字典match，字典的key为像素，value为聚类中心。字典存储所有已经配对的像素与聚类中心。

图片包含 屏幕截图

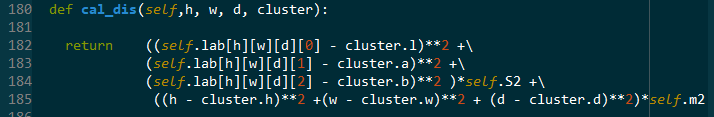
描述已自动生成

1. assign方法对所有聚类中心的2S\*2S\*2S领域内所有的像素点进行遍历。代码的163和165行是防止2S\*2S\*2S领域内的点超出像素范围。如果像素点[h,w,d]与聚类中心cluster[k]的距离小于distance[h][w][d], 则需要更新distance[h][w][d]的值。如果像素（h,w,d）在字典match的Key list中，说明（h,w,d）之前与其他的聚类中心配对过，需要除去之前与该像素配对的cluster的pixels里的该像素（代码172,173行）。之后将新的键值对添加到match字典中。最后把像素（h,w,d）加入到cluster的pixels列表中。

此处有字典match和列表cluster.pixels都记载了配对信息，这样做的好处是：当更新cluster时，只需要遍历cluster.pixels中的pixel，计算平均值，即可得到新的cluster。而match的作用是当需要更新与某个像素匹配的聚类中心时，可以通过match字典获得上一次与该像素匹配的聚类中心，由此去除该聚类中心的piexels列表中的该像素。



1. dal\_dis方法，计算像素（h,w,d）与cluster的距离。



1. renew\_cluster方法，更新聚类中心。

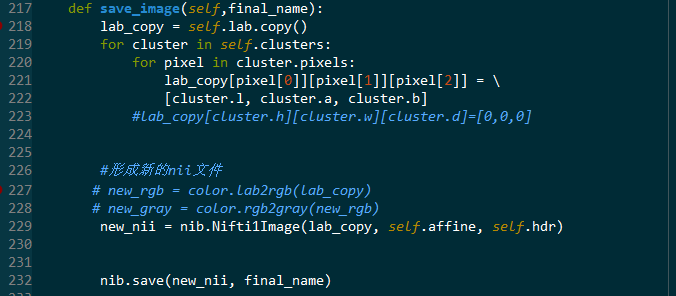
图片包含 文字

描述已自动生成

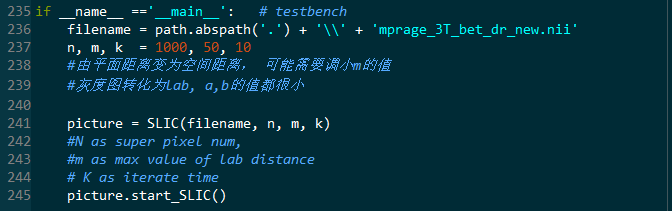
1. 代码218行把self.lab的值拷贝到了lab\_copy,因为self.lab在下一次迭代时还需要使用，不能更改。所以更改lab\_copy里像素的lab值并输出为图片。

将每个聚类中心的cluster.pixels列表中的全部像素的lab值改为cluster的lab值，并把cluster的lab值设为（0,0,0），因此cluster在输出图片中将以黑点的形式显现。

最后用之前记录的图像的*仿射矩阵、头文件和新的lab数据生成Nii图像，*调用nib.save方法输出图片到指定路径。



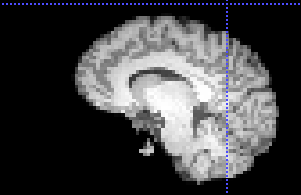
1. 主函数通过引用了相对路径（被执行的py文件所在目录），只需要把需要处理的图片放入py文件所在目录，即可以运行程序。即使文件改变了所在位置，也不需要对路径进行更改。

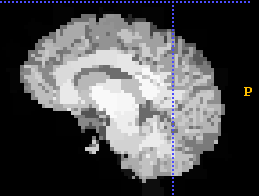


## 程序的运行结果

设置超像素个数n=1000, LAB距离最大值m=10，迭代次数为10，像素为85\*61\*58的图片迭代一次用时约18秒,迭代十次总共用时211秒。

Depth=29（即3D图像的第三个维度取值为29）时的平面图为：

图片包含 动物, 软体动物, 无脊椎动物

描述已自动生成

原图 k=1 k=10

Height=50（即3D图像的第一个维度取值为50）时的平面图为：

图片包含 无脊椎动物, 动物, 软体动物

描述已自动生成图片包含 动物

描述已自动生成图片包含 动物, 无脊椎动物, 软体动物

描述已自动生成

原图 k=1 k=10

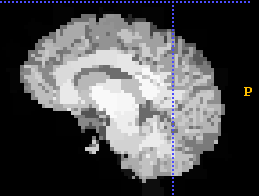
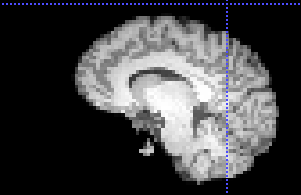
# 对SLIC分割SperVoxel的认识和思考

## 参数m取值范围的变化

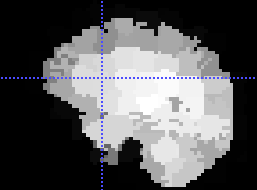
由于由平面距离变为空间距离，Ds的取值范围变大，而Dc的取值范围与2维时相同，因此应该考虑减小常数m的取值范围，以保持Ds对距离函数D的影响所占比例不变。把m的最大值减小为最大值的2/3, 把m的取值范围由【1,40】减小为【1，25】。

在之前的报告中，处理二维图像时，m取值为30往往能取得较好的分割效果。但在三维图像中，我们可以看看以下的一组对比图。

Depth=29（即3D图像的第三个维度取值为29）时的平面图为：



原图 m=10( 迭代10次)



M=30(迭代10次)

显然，m=30时SuperVoxel的块状感太强，与原图对比失真非常严重，而m=10时分割效果比较好。由此得出以上缩小m取值范围的做法是合理的，建议处理三维灰度图像时,m取值为10.(处理三维彩色图像时m的最佳取值可能与处理三维灰度图像有差别，但总的来说也应该小于30)

## 灰度gray与亮度L

本文中的3D医学图像样例是灰度图而非彩色图，通过nibabel.load.get\_data方法获取数据后得到的是灰度值的三维numpy矩阵。而参照之前报告的思路，应该采用LAB色彩来计算距离。因此笔者通过调用skimage.color中的gray2rgb和rgb2lab来完成灰度到LAB空间的转换。

但转换后笔者观察存储有LAB的四维numpy矩阵，发现关于矩阵中A，B的值都非常地小，很接近零，如下图。

图片包含 墙壁

描述已自动生成

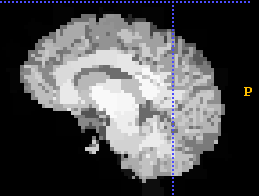
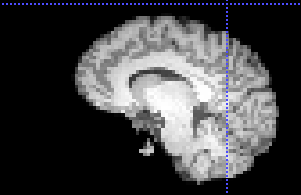
其实这是十分显而易见的事情，因为LAB本就是由灰度图转化而成的，而灰度图只有灰度这一个信息，因此转化为LAB图后，也应该基本上只有亮度一个信息（虽然灰度和亮度并不完全等价），而代表颜色的A和B信息基本为0.

查阅资料可知，灰度gray是关于rgb的线性函数，而亮度L是关于rgb的分段函数（两段分别是二次和三次函数）。经过简单的数学推导，在灰度图（A和B为0，或者说rgb三者取同一个值时）时，灰度gray与亮度L具有相同的单调性。

而在灰度图片中，灰度比LAB更容易获得，不需要额外调用转化函数。而且灰度只有一个维度，而LAB有三个维度需要计算，因此若能用灰度代替LAB计算距离，不仅能简化程序，而且大大地提升程序的运行速度。

由于亮度L的取值范围为【0，100】，而灰度gray的取值范围为【0，255】，因此如果用灰度代替亮度计算距离D，应该增大m,m本来取值为10，此处取值为

Depth=29（即3D图像的第三个维度取值为29）时的平面图为：



原图 采用LAB，m=10( 迭代10次)

图片包含 动物, 无脊椎动物, 软体动物

描述已自动生成

采用gray，m=30(迭代10次)

由上图知，采用gray比采用LAB分割SuperVoxel的效果略差一些，但是采用LAB为85\*61\*58的图片迭代一次用时约18秒,迭代十次总共用时216秒，而采用gray为85\*61\*58的图片迭代一次用时约12秒,迭代十次总共用时131秒，比用LAB快了将近一倍。（采用gray的SLIC算法的python实现见附件，命名为SuperVoxel\_gray.py；采用LAB的SLIC算法的python实现见附件，命名为SuperVoxel.py）

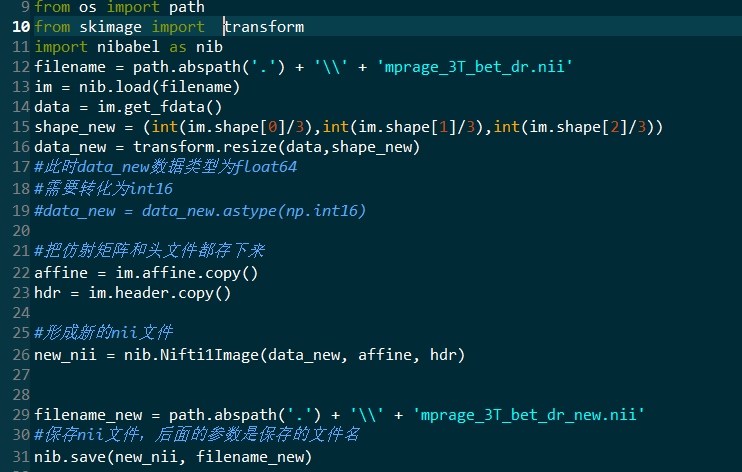
因此，若要求更快的分割速度，同时对分割效果的要求较低时，采用gray代替LAB分割灰度图不失为一种选择。但是需要记住要更改m的取值，并且此方法只适用于灰度图，不适用于彩色图。

## 压缩3D图像

用SLIC分割2D图像时，时间复杂度为O(k\*N), 其中k迭代次数， N是所有像素的个数。同理分析，用SLIC分割3D图像时，时间复杂度仍为O(k\*N)。但由于3D图像比2D图像具有更多的像素点，因此处理起来更为耗时。由于笔者的笔记本电脑计算速度较慢，因此先对3D图像进行了压缩处理，再进行SuperVoxel分割。

压缩程序的代码如下（压缩程序见附件，命名为condense\_img.py）：

15，16行通过调用skimage.transform,把原3D图像的长宽高都压缩为原来的1/3.



## 修改数据类型

本次实验中采用的存储LAB数据的numpy矩阵的数据类型是float64，而实际上存储LAB用int16就可以，只是精度稍低。而且计算机对整数比浮点数的计算快很多，因此可以采用numpy.astype方法，把LAB矩阵里的float64改为int16.

但笔者并未在本程序中实现这一点，待以后有时间再尝试。

# 参考文献

1. Radhakrishna Achanta, et al. "SLIC superpixels compared to state-of-the-artsuperpixel methods." Pattern Analysis and Machine Intelligence, IEEETransactions on 34.11 (2012): 2274-2282.

【2】ITK-SNAP为一个医学图像处理相关的开源软件。

下载网址为<http://www.itksnap.org/pmwiki/pmwiki.php?n=Downloads.SNAP3>