基于 Mean-Shift 的图像分割

刘嘉伟

Hefei University of Technology

2020 年 11 月 6 日

摘要

图像分割技术是计算机视觉领域的个重要的研究方向，是图像语义理解的重要一环。图像分割是指将图像分成若干具有相似性质的区域的过程，从数学角度来看，图像分割是将图像划分成互不相交的区域的过程。 本文简要概述了目前流行的各种图像分割算法，并详细介绍了通过传统方法—基于均值漂移的图像分割算法，将一幅 RGB 三通道图像进行分割，并对算法及分割结果进行评价。

关键字：计算机科学与技术，机器视觉，图像分割，均值漂移

Abstract

Image segmentation technology is an important research direction in the field of computer vision and an important part of image semantic understanding. Image segmentation refers to the process of dividing an image into several areas with similar properties. From a mathematical point of view, image segmentation is the process of dividing an image into disjoint areas. This paper introduces the traditional method—an image segmentation algorithm based on mean shift to segment an RGB three-channel image, and evaluate the algorithm and segmentation results.

Keywords: CS ，Image Segementation，Mean-Shift，Computer Vision

1 图像分割算法概述

这一大部分我们将要介绍的是深度学习出现之前人们利用数字图像处理、拓扑学、数学等方面的只是来进行图像分割的方法。当然现在随着算力的增加以及深度学习的不断发展，一些传统的分割方法在效果上已经不能与基于深度学习的分割方法相比较了，但是其中的思想以及那些数学理论的推导还是很值得我们借鉴于学习的。

基于阈值的分割方法

灰度阈值分割法是一种最常用的并行区域技术，它是图像分割中应用数量最多的一类。

$$g(x,y)=\begin{cases}1, f(i,j)\geq T\\ 0, f(i,j)<T\end{cases}$$

其中，T为阈值；对于物体的图像元素，$\mathrm{g(i,j)}=1$，对于背景的图像元素，$\mathrm{g(i,j)}=0$

由此可见，阈值分割算法的关键是确定阈值，如果能确定一个合适的阈值就可准确地将图像分割开

来。阈值确定后，将阈值与像素点的灰度值逐个进行比较，而且像素分割可对各像素并行地进行，分割的结果直接给出图像区域。

阈值分割的优点：是计算简单、运算效率较高、速度快。人们发展了各种各样的阈值处理技术，包括全局阈值、自适应阈值、最佳阈值等等。

边缘分割

图像分割的一种重要途径是通过边缘检测，即检测灰度级或者结构具有突变的地方，表明一个区域的终结，也是另一个区域开始的地方。这种不连续性称为边缘。不同的图像灰度不同，边界处一般有明显的边缘，利用此特征可以分割图像。图像中边缘处像素的灰度值不连续，这种不连续性可通过求导数来检测到。

对于阶跃状边缘，其位置对应一阶导数的极值点，对应二阶导数的过零点(零交叉点)。因此常用微分算子进行边缘检测。常用的一阶微分算子有 Roberts 算子、Prewitt 算子和 Sobel 算子，二阶微分算子有 Laplace 算子等。在实际中各种微分算子常用小区域模板来表示，微分运算是利用模板和图像卷积来实现。这些算子对噪声敏感，只适合于噪声较小不太复杂的图像。

直方图法

基于直方图的方法是非常有效的图像分割方法，因为他们通常只需要一个通过像素。在这种方法中，直方图是从图像中的像素的计算，并在直方图的波峰和波谷是用于定位图像中的簇。颜色和强度可以作为衡量。这种技术的一种改进是递归应用直方图求法的集群中的形象以分成更小的簇。重复此操作，使用更小的簇直到没有更多的集群的形成。

基于区域生长的分割方法

(1)区域生长

区域生长的基本思想是将具有相似性质的像素集合起来构成区域。具体先对每个需要分割的区域找一个种子像素作为生长的起点，然后将种子像素周围邻域中与种子像素有相同或相似性质的像素（根据某种事先确定的生长或相似准则来判定）合并到种子像素所在的区域中。将这些新像素当作新的种子像素继续进行上面的过程，直到再没有满足条件的像素可被包括进来。这样一个区域就长成了。 例如：分水岭算法，浸水填充。

(2)区域分裂合并

区域生长是从某个或者某些像素点出发，最后得到整个区域，进而实现目标提取。分裂合并差不多是区域生长的逆过程：从整个图像出发，不断分裂得到各个子区域，然后再把前景区域合并，实现目标提取。分裂合并的假设是对于一幅图像，前景区域是由一些相互连通的像素组成的，因此，如果把一幅图像分裂到像素级，那么就可以判定该像素是否为前景像素。当所有像素点或者子区域完成判断以后，把前景区域或者像素合并就可得到前景目标。基于图的分割其实也是这种思想。

不足：基于区域的分割方法往往会造成图像的过度分割

基于图的图像分割

图像的图表示是指将图像表达成图论中的图。具体说来就是，把图像中的每一个像素点看成一个顶点 vi ∈ V（node 或 vertex），像素点之间的关系对（可以自己定义其具体关系，一般来说是指相邻关系）构成图的一条边 ei ∈ E，这样就构建好了一个图 G = (V,E)。图每条边的权值是基于像素点之间的关系，可以是像素点之间的灰度值差。

图是由顶点集 V（vertices）和边集 E（edges）组成，表示为 G=(V, E)，每个像素点代表图的一个顶点 vi ∈ V ，相邻的两个像素点构成一条边(vi, vj) ∈ E, 像素颜色值的差异构成边(vi, vj)的权值 w((vi, vj))，初始化时每一个像素点都是一个顶点，然后逐渐合并得到一个区域，确切地说是连接这个区域中的像素点的一个 MST。

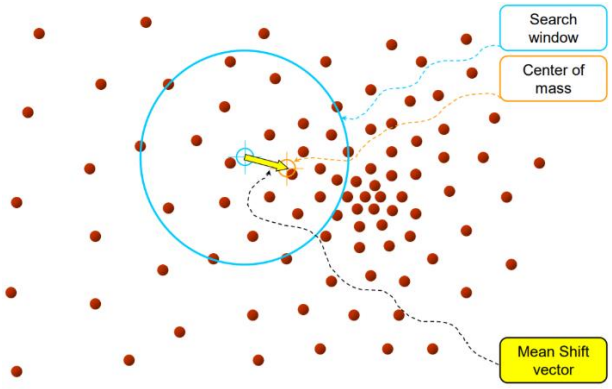
- 最小生成树（MST）：给定需要连接的顶点，选择边权之和最小的树。

基于数学形态学的分割方法

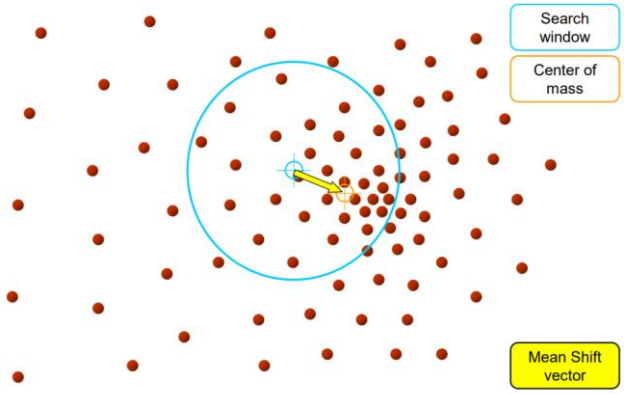
数学形态学是一种非线性滤波方法，可以用于抑制噪声、特性提取、边缘检测、图像分割等图像处理问题。数学形态学首先被用来处理二值图像，后来也被用来处理灰度图像，数学形态学的特点是能将复杂的形状进行分解，并将有意义的形状分量从无用的信息中提取出来。

二　Mean-Shift 算法原理

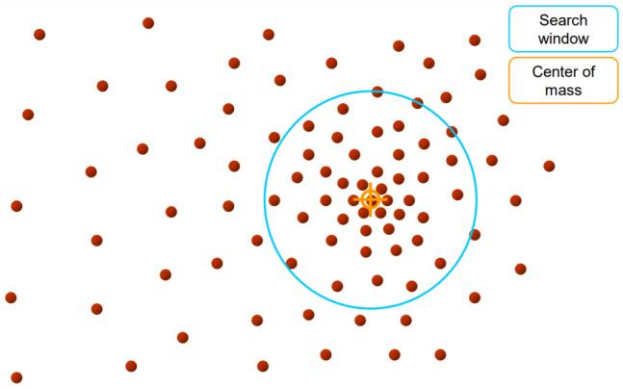
Mean-shift 算法，中文译作均值漂移算法，其核心思想就是，假如在一个二维平面上分布着许多的点，每一个区域的点的密度都不一样，如下图所示：



均值漂移算法是对图像上的每个点进行如下处理，假设当前处理的点是x：选定一个搜索半径 R，对距离x周围 R 范围内的点，计算所有点x_i的平均位置向量，$\frac{1}{k}\sum x - x_i$，其中 k 为半径 R 范围内点的数量，然后将点x漂移至平均位置，如下图：



然后重复上述操作直至收敛，最终结果如下：



以上便是 mean-shift 算法的基本原理，但是如何将该算法用于图像分割呢？下面我们将该算法应用到图像分割中，变成一个图像分割算法。

三　Mean-Shift 图像分割算法

图像中的点x包括两类信息：坐标空间（spatial，x^s，$[px, py]$），颜色空间（range，x^r，$[r, g, b]$）。这些就构成了特征空间。

模点搜索：某一个点x，它在联合特征空间px, py, r, g, b中迭代搜索它的 mode/模点y；

图像平滑: 将模点的颜色值赋给它自己，即$x^r = y^r$.对应原论文文中的**图像平滑，**实质上是**通过模点搜索，达到图像平滑的效果。**

设点x_i依次爬过的脚印为：

$$\{y_{i,0}, y_{i,1}, y_{i,2}, \dots, y_{i,k} \dots y_{i,c}\}$$

出发时$y_{i0} = x_i$，它所收敛到的模点为$y_{i,c}$，c 代表 convergence。

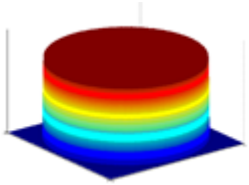
第一步：如果迭代次数超过最大值（默认最多爬最多 100 次），结束搜索跳到第四步，否则，在坐标空间，筛选靠近$y_{i,k}^s$的数据点进入下一步计算。

第二步：使用第一步幸存下来的点计算重心，并向重心移动。

$$y_{i,k+1}^S = \frac{\sum_{n=1}^{N} x_n^s g(||\frac{x_n^r - y_{i,k}^r}{h_r}||^2)}{\sum_{n=1}^{N} g(||\frac{x_n^r - y_{i,k}^r}{h_r}||^2)}$$

其中g是某种核函数，比如高斯分布，h_r是颜色空间的核平滑尺度。OpenCV 内置函数使用的是最简单的均匀分布：

$$g(x)=\begin{cases}1\ x\le 1\\0\ otherwise\end{cases}$$



二维可视化效果

$g(|\left|\frac{x_n^r-y_{i,k}^r}{h_r}\right|^2)$是一个以$y_{i,k}^r$（第$i$步位置的颜色值）为球心，半径为$h_r$的球体，球体内部值为 1，球体外部值为 0。对于经过上一步筛选后幸存的数据点x_n，如果其颜色值x_n^r满足$||x_n^r-y_{i,k}^r<h_r||$，也就是颜色值落也在球内，那么求重心$y_{i,k+1}^s$时，就要算上x_n^r，否则落在球外，算重心时，就不带上它。实际上，上一步是依据空间距离筛选数据点，g是依据颜色进一步筛选数据点，上一步的筛子是圆形，这一步的筛子是球体。

四　后续处理步骤

在使用 Mean-Shift 算法处理图像后，我们已经将图像的每个像素点进行了均值漂移，但是这并不足以完成我们需要的图像分割达到的效果，我们需要进行进一步处理，这里我使用的是**合并相似区域/模点聚类**算法，其核心思想就是，从图像上一个点开始，如果和它附近的点（4/8 邻域）的颜色值相似就合并，同时再从新合并的点出发继续合并下去，直到碰到不相似的点或者该点已经属于另一类了，此时，就退回来，直到退无可退（所有的 4/8 邻域搜索空间都已经搜索完毕）。

在这步处理结果后，还可能有一些小区域没有被合并起来，可能会影响效果，所以还可以选择性的进行**兼并小区域**，从一没有合并到大区域的小区域出发，寻找最近的大区域，将其像素值合并到大区域中。

简而言之，设满足$||x_n^r-y_{i,k}^r<h_r||$的点依次为$\{n_1,n_2,\dots,N_k\}$，那么重心计算公式可以进一步化简为：$y_{i,k+1}^s=\frac{\sum_{n=1}^{N_k}x_n^s}{N_k}$

注意：上文中的两个参数h_r,h_s，是 Mean shift 最核心的两个参数，具有直观的意义，分别代表坐标空间和颜色空间的核函数带宽。

第三步：判断是否到模点了，到了就停止。如果，移动后颜色或者位置变化很小，则结束搜索，跳到第四步，否则重返第一步，从y_{k+1}继续爬。

OpenCV 内置函数停止搜索的条件：

（1）坐标距离不变$y_{i,k+1}^s==y_{i,k}^s$

（2）颜色变化值很小，$||y_{i,k+1}^r-y_{i,k}^r||\le thr$。

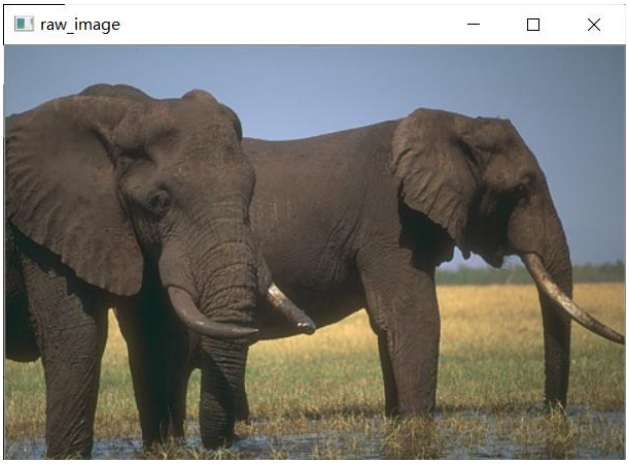
满足一条就可以功成身退，否则继续努力。

第四步：将模点$y_{i,c}$的颜色$y_{i,c}^r$赋给出发点$x_i/y_{i,0}$，即$x_i^r\leftarrow y_{i,c}^r$。

五　结果分析

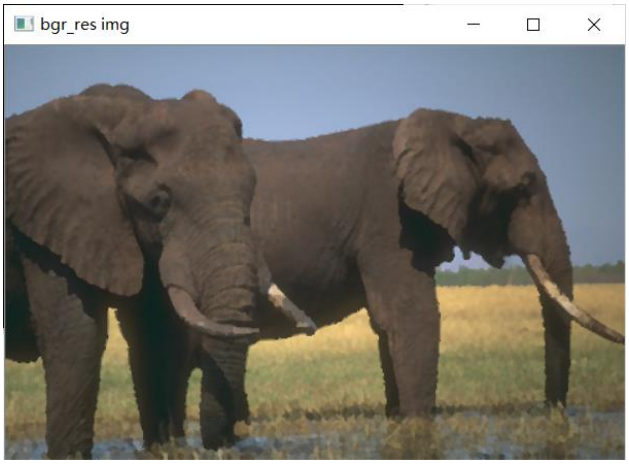
5.1　结果演示

待分割的原图像如图所示：



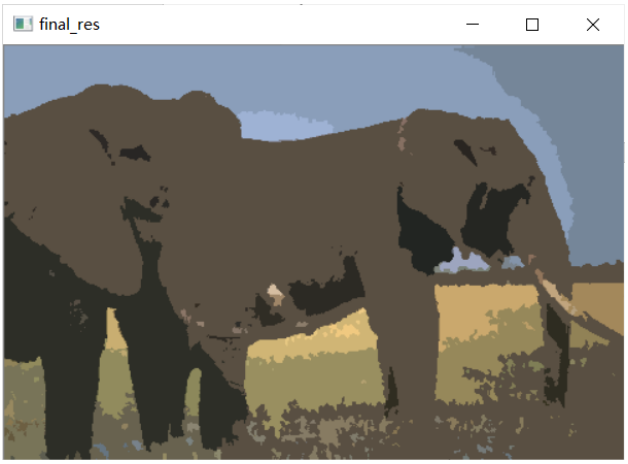
原始待图像

经 Mean-Shift 算法处理后的图像：



Mean-shift 处理图

可以看见，每个像素点都有一定的变化，在一定区域内的像素点都漂移到同一个地方，也就是说在一小块区域中像素值都是一样的了，但是这个结果我们并不满意，还需要进行进一步的处理，也就是合并相似区域。

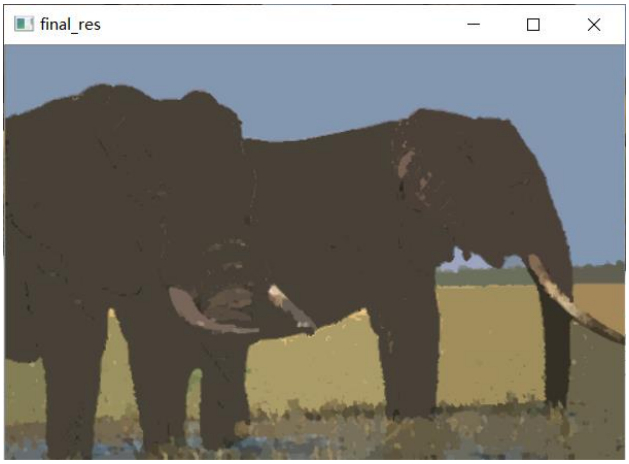


mergeThreshold = 60

填充结果：

填充图像

该图像每种颜色都是一块区域。上述的填充结果是从一个像素点开始扩展，在扩展的过程中都是以开始扩展的点作为参照点进行合并，所以可以看到大象上面的天空被分成了三层，结果可能还有点瑕疵，所以，我还扩展了一种合并方案，就是在扩展过程中，变扩展边动态的将参照点变成当前出队列的点，这样就会分的更明显，相同的区域会分为同一块，但是此时合并像素的阈值要调整的小得多，一般在 10 以内，因为如果是动态调整参照像素，则邻近的两个像素颜色变化是很小的，如果稍微调高了一点，那么整幅图都会被分为同一块，效果十分差劲，经过我多次调参尝试，最佳的参数范围在 5~8piexl 的距离，这种模式只需要修改一个参数，然后调整合并阈值。

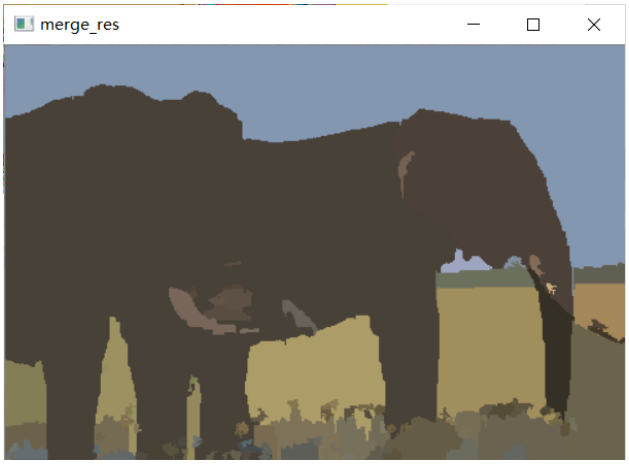


动态更新参考点结果

mergeThreshold = 8

动态更新参考点填充图像

我们会发现，在这种模式下，会产生很多未被合并的小区域图像，使得结果看起来非常糟糕，所以，图像还需要进行进一步的处理，即兼并小区域。兼并小区域的思想就是，在上一步 bfs 合并相似区域的过程中，记录下了每个区域的像素点个数，然后我们再次通过遍历图像，找到区域中的像素个数少于 lowThreshold 的区域，然后再次通过 bfs 寻找到周围的像素个数大于 hiThreshold 的像素区域，将该区域合并进大区域，就实现了兼并小区域。

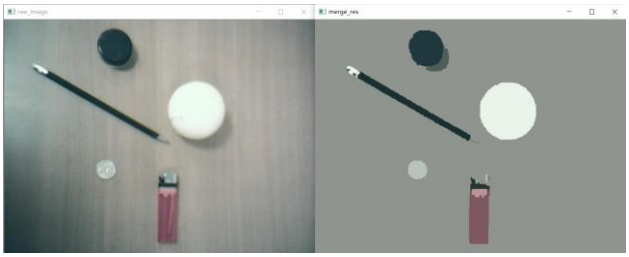


小区域兼并分割图

小区域兼填充图

可以看到，最后的分割结果十分理想。

七　结果展示

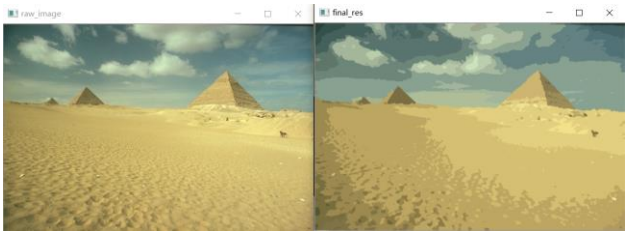


分割前后对比图

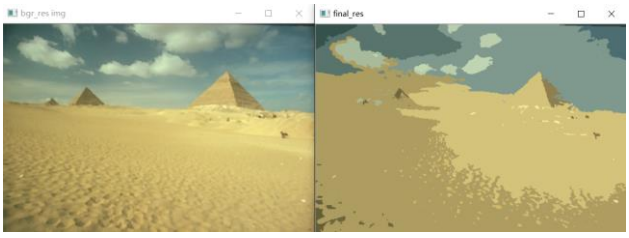
分割填充图

八　算法评价

Mean-Shfit 算法的时间复杂度较高，需要将每个像素点遍历一次，然还还需要迭代 T 次，假设图片的大小为$O(MN)$，那么其时间复杂度为$O(\mathrm{T}*\mathrm{MN})$，在遍历每个点时，还需要进行圆形范围的搜索，我是通过圆的外接正方形来寻找圆形区域所以时间复杂度应修正为$O(T*MN*r^2)$,搜索半径 r 对该算法时间的影响非常关键，当 r 设为 8 时，每张图片的平均处理时间约为 6 秒，当 r 设为 16 时，每张图片的平均处理时间达 20 秒，耗时较多。对合并像素的阈值 colorThreshold 的设置也很关键，如果想要将图像分的细一些，那么最好设置为 20~30，如果想将图像分割的部分少一些，那么需要设置为 50~80。



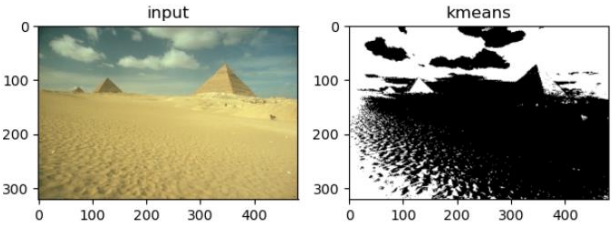
colorThreshold=30



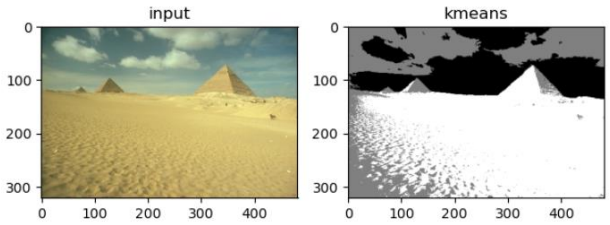
colorThreshold=65

与 k-means 算法比较：

k-means 算法是通过聚类的无监督学习来进行图像分割，需要指定聚类的中心点数，该数就是需要进行分割出的图像的结果类数，因为该算法是通过无监督学习，每次聚类中心的位置都是随即指定，所以该算法的结果不可预知，并且调试也不方便



K=2 的 k-means



K=3 的 k-means

从上述结果来看，个人感觉基于 k-means 分割的效果不如 mean-shift 的好，但是 k-means 的时间复杂度远比 mean-shift 要低，不需要一秒就能得到一个图像分割的结果，k-means 时间复杂度为 $O(knT)$，其中 k 是聚类中心个数，n 是样本点数，T 是迭代次数。