Efficient Graph-Based Image Segmentation

和树繁

Sofan_He@163.com

计算学部
哈尔滨工业大学

11th April 2022

任务介绍

- 本文所处理的是图像分割问题, 对应到目前深度学习的普通分割. 区别于语义分割和实例分割, 普通分割需要将所有的部分均分割出来, 而不关注不同部分之间是否存在语义上的相关性.

任务介绍

- 本文所处理的是图像分割问题，对应到目前深度学习的普通分割. 区别于语义分割和实例分割，普通分割需要将所有的部分均分割出来，而不关注不同部分之间是否存在语义上的相关性.
- 本文的目标是开发出一种**可计算的**图像分割方法，且该方法能够像边缘检测一样被广泛应用. 为此，该方法需要满足如下属性:

任务介绍

- 本文所处理的是图像分割问题，对应到目前深度学习的普通分割. 区别于语义分割和实例分割，普通分割需要将所有的部分均分割出来，而不关注不同部分之间是否存在语义上的相关性.
- 本文的目标是开发出一种**可计算的**图像分割方法，且该方法能够像边缘检测一样被广泛应用. 为此，该方法需要满足如下属性:
  - 捕捉视觉感知上重要的区域. 对应算法能够映射出图片的全局特征.
  - 计算复杂度方面与图像像素个数成近似线性关系.

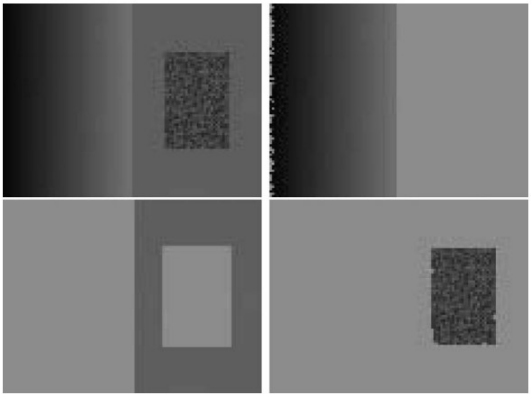
相关工作

- 本文提出的方法与其他类别分割方法的不同.
  - 基于特征向量的分割方法. 这种方法计算复杂性太高, 无法进行广泛应用.
  - 其他计算有效的方法往往无法捕捉全局视觉特征.
  - 本文的方法可以捕捉全局视觉特征, 且可以以 $O(n\log n)$ 的复杂度完成 n 个像素点的分割.

相关工作

- 本文提出的方法与其他类别分割方法的不同.
  - 基于特征向量的分割方法. 这种方法计算复杂性太高, 无法进行广泛应用.
  - 其他计算有效的方法往往无法捕捉全局视觉特征.
  - 本文的方法可以捕捉全局视觉特征, 且可以以 $O(n\log n)$ 的复杂度完成 n 个像素点的分割.
- 本文提出的方法与其他基于图的分割方法的不同.
  - 分割边界的判定标准是自适应调整的.
  - 该调整策略是计算复杂性可接受的, 而其他衡量策略为 NP-hard 问题.

一个例子



- 左上是原图，其他三个图片为本文提出的算法分割出的三个不同的区域.
- 该图像三个区域，左侧区域是渐变区域，右侧外部为恒值区域，右侧中间为高方差区域.
  - 第一个挑战是，高方差区域如何能够被分割为同一个区域.
  - 第二个挑战是，不能使用全局统一的阈值来进行分割.

问题定义

- 定义无向图 $G = (V, E)$, 其中 $v \in V$ 为原图中的一个像素点, $(v_i, v_j) \in E$ 对应了一对相邻的像素点.
- 在 E 上定义一个函数 $w(v_i, v_j)$ 表示一个非负的不同像素点之间的差异度.
- 定义一个分割 S 为: 将原图分为了若干区域 $C \in S$, 每个区域对应生成子图 $G' = (V, E'), E' \subseteq E$ 中的一个连通区域.

分割标准

- 定义函数 D 为两个区域间是否存在边界的判定函数, 即分割标准. 该标准基于边界元素的差异性计算得出.
- 定义一个区域 $C \subseteq V$ 的内部差异 $Int(C)$ 为该区域最小生成树 $MST(C, E)$ 的最大边权. 即

$$Int(C) = \max_{e \in MST(C,E)} w(e) \tag{1}$$

- 定义两个区域 $C_1, C_2 \subseteq V$ 的区域间差异 $Dif(C_1, C_2)$ 为连接两个区域的边的最小权值. 若不存在相连的边则 $Dif(C_1, C_2)$ 取值为 ∞. 其计算公式如下.

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w(v_i, v_j) \tag{2}$$

分割标准

- 两个区域之间存在边界的判定标准为，两个区域之间的差异 $Dif(C_1, C_2)$ 大于至少一个区域的内部差异 (同时也应该大于人为给定的某个阈值). 即

$$D(C_1, C_2) = \begin{cases} true, & \text{if} Dif(C_1, C_2) > MInt(C_1, C_2) \\ false, & \text{otherwise} \end{cases} \tag{3}$$

- 其中表示最小内部差异的函数 $MInt$ 定义如下:

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2)) \tag{4}$$

- 其中 $\tau(C)$ 表示一个人工阈值，为了避免 $|C| = 1, Int(C) = 0$ 时被随意合并的情况. 同时，该边界函数也可以用来控制分割区域的形状的信息. 文中给出了一个以大小为参考的样例函数 $\tau(C)$ 如下:

$$\tau(C) = k/|C| \tag{5}$$

Segmentation Algorithm

Input: $G = (V, E), |V| = n, |E| = m$

Output: $S = (C_1, C_2, C_3, ..., C_r), C_i \subseteq V$

Sort E into $\pi = (o_1, ..., o_m)$, by non-decreasing edge weight ;

Start with a segmentation S^0, where each vertex v_i is in its own component ;

Repeat step 3 for $q = 1, ..., m$;

Construct S^q given S^{q-1} as follows. Let v_i and v_j denote the vertices connected by the q-th edge in the ordering, i.e., $o_q = (vi, vj)$. If v_i and v_j are in disjoint components of S^{q-1} and $w(o_q)$ is small compared to the internal difference of both those components, then merge the two components otherwise do nothing. More formally, let C_i^{q-1} be the component of S^{q-1} containing v_i and C_j^{q-1} the component containing v_j. If $C_i^{q-1} \neq C_j^{q-1}$ and $w(o_q) \leq MInt(C_i^{q-1}, C_j^{q-1})$ then S^q is obtained from S^{q-1} by merging C_i^{q-1} and C_j^{q-1}. Otherwise $S^q = S^{q-1}$;

Return $S = S^m$.

分割算法的核心思路

- 初始化时将所有点均视为独立的集合. 将边按照权值非降的顺序排序.
- 枚举排序后的每条边, 判定该边连接的两个点所在的区域是否存在边界.
  - 若两个点在同一个区域, 则跳过.
  - 若两个点不在同一个区域, 且 $D(C_1, C_2) = \mathit{false}$, 则将两个区域进行合并.
  - 若两个点不在同一个区域, 且 $D(C_1, C_2) = \mathit{true}$, 则什么都不做.
- 返回枚举所有边处理后的结果.

分割算法的性质

- 定义 1. 分割 S 太细致为: 存在一对区域 $C_1, C_2 \in S$ 但不能证明其间存在边界.
- 定义 2. 分割 S 太粗糙为: 存在一个更细化的分割满足其不是太细致的分割.
- 性质 1. 对于任何 $G = (V, E)$, 存在很多分割 S 既不是很细致的也不是很粗糙的.
- 引理 1. 若对于某次枚举 $o_q = (v_i, v_j)$, 其所属的两个区域 C_i, C_j 没有被合并. 则 C_i, C_j 至少有一个区域会出现在最后的分割 S 中.
- 定理 1. 算法所得的分割 S 不是太细致的分割.
- 定理 2. 算法所得的分割 S 不是太粗糙的分割.
- 定理 3. 算法分割结果并不依赖非递减排序的选择.

算法实现与复杂度计算

- 记总边数为 m 条. 算法可以被分成将不同边排序和重复判定是否可以合并两部分.

算法实现与复杂度计算

- 记总边数为 m 条. 算法可以被分成将不同边排序和重复判定是否可以合并两部分.
- 算法第一部分使用任何排序算法都可以达到 $O(m \log m)$.

算法实现与复杂度计算

- 记总边数为 m 条. 算法可以被分成将不同边排序和重复判定是否可以合并两部分.
- 算法第一部分使用任何排序算法都可以达到 $O(m \log m)$.
- 算法第二部分与 Kruskal 算法非常相似, 函数 $Int(C)$ 可以在常数时间内完成合并, $Diff(C_1, C_2)$ 函数为每次枚举到的最小的连接的边, 因此算法整体复杂度为 $m\alpha(m)$, 其中 α 为路径压缩并查集中的 Ackerman 函数 (增长非常慢, 近似线性).

算法实现与复杂度计算

- 记总边数为 m 条. 算法可以被分成将不同边排序和重复判定是否可以合并两部分.
- 算法第一部分使用任何排序算法都可以达到 $O(m \log m)$.
- 算法第二部分与 Kruskal 算法非常相似, 函数 $Int(C)$ 可以在常数时间内完成合并, $Diff(C_1, C_2)$ 函数为每次枚举到的最小的连接的边, 因此算法整体复杂度为 $m\alpha(m)$, 其中 α 为路径压缩并查集中的 Ackerman 函数 (增长非常慢, 近似线性).
- 官方给出的代码仓库为`http://cs.brown.edu/people/pfelzens/segment`. 我实现了单通道 (目前选择的是 R 通道, 对于灰度图而言是一样的) 不带高斯平滑的基于邻域的分割版本. 仓库地址`https://gitee.com/Sofan_He/efficient-graph-based-image-segmentation-reproduce`.

实现细节

- 边由原图中空间坐标的 8-邻域计算得来，边权为灰度像素值的绝对值差 $w(v_i, v_j) = |I(p_i) - I(p_j)|$. 使用高斯滤波 $\delta = 0.8$ 对图像进行平滑处理.
- 阈值函数选择使用 $\tau(C) = k/|C|$. 对于 COIL 数据集中 128x128 的图像使用 $k = 150$, 对于 320x240 的街景图像使用 $k = 300$.
- 上述算法只能处理单通道图像的情况，对于彩色 RGB 通道的图像:
  - 分别对 RGB 三个通道进行处理，然后求分割结果的交集.
  - 定义距离函数为颜色空间的距离. 进行单次分割.

  作者在论文中表示，三个通道分别处理的方法效果更好.

实验结果 1

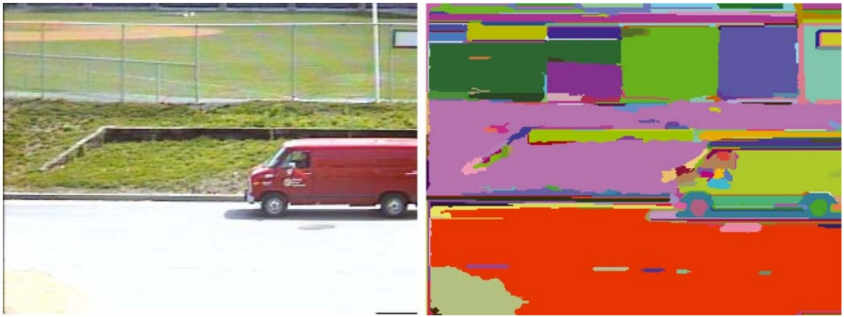


Figure 2. A street scene (320 × 240 color image), and the segmentation results produced by our algorithm ($\sigma = 0.8, k = 300$).

图中草地部分变化较大，而本文所提出的算法可以很好的处理这种内部差异大的情况.

实验结果 2

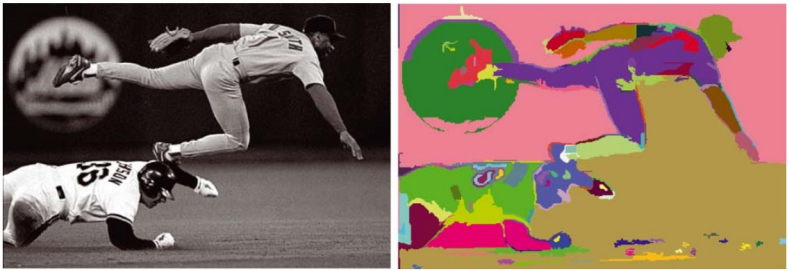


Figure 3. A baseball scene (432 × 294 grey image), and the segmentation results produced by our algorithm ($\sigma = 0.8, k = 300$).

作者认为草地和背景被分为同一类是因为在图中没有明显的分割界限.

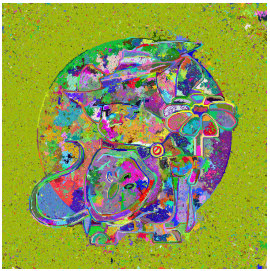
实验结果 3

Figure 4. An indoor scene (image 320 × 240, color), and the segmentation results produced by our algorithm ($\sigma = 0.8$, $k = 300$).

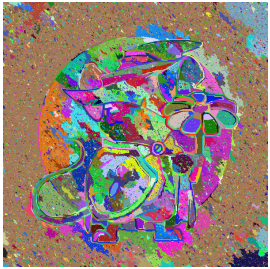
从这个室内复杂场景来看，算法能够分割出小的特征区域.

实现效果对比

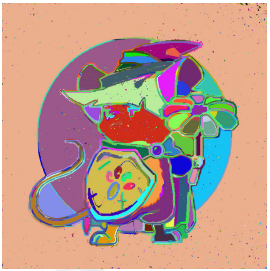
原图，在放大看的条件下能够看到非常多的噪声



官方代码的分割结果，$\sigma = 0, k = 300, min = 1$



我的代码的分割结果，参数设置在代码中固定且同左



官方代码的分割结果，$\sigma = 0.8, k = 300, min = 1$

实现细节

- 边由原图中每个像素点的特征空间距离计算得出，对于每个点连接距离最近的 N 个点. 不同点 v_i, v_j 之间的距离计算公式为:

$$w(v_i, v_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (r_i - r_j)^2 + (g_i - g_j)^2 + (b_i - b_j)^2} \quad (6)$$

- 阈值函数选择使用 $\tau(C) = k/|C|$. 对于 COIL 数据集中 128x128 的图像使用 $k = 150$, 对于 320x240 的街景图像使用 $k = 300$.
- 其中寻找最近点使用 *ANN*(Arya and Mount, 1993) 算法实现. 其他实现方式同上.
- 这种方法能够捕捉到更多的全局特征，而不必仅仅局限在空间邻域里.

实验结果 1

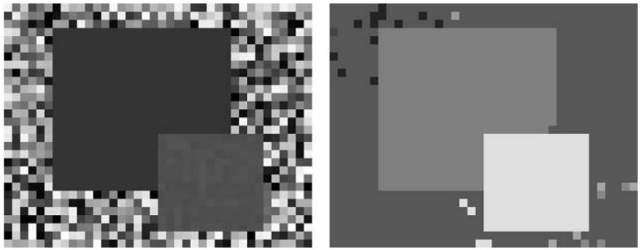
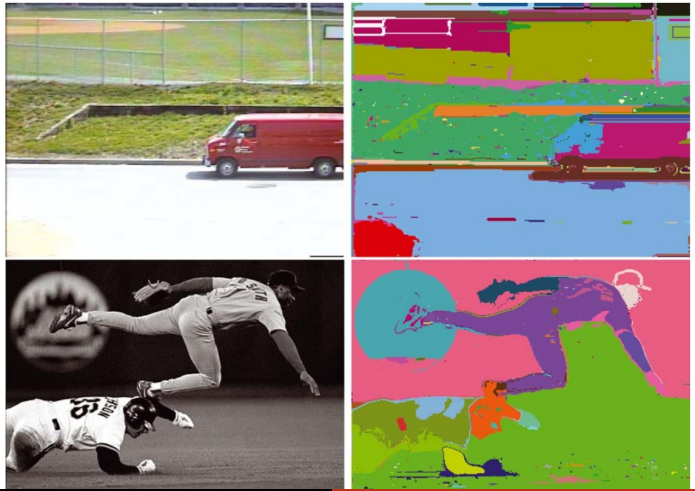


Figure 6. A synthetic image (40×32 grey image) and the segmentation using the nearest neighbor graph ($\sigma = 0, k = 150$).

实验结果 2



实验结果 3

Figure 8. Segmentation using the nearest neighbor graph can capture spatially non-local regions ($\sigma = 0.8$, $k = 300$).

从该图可以看出使用特征空间近邻的方法可以更好的捕捉到非局部的特征.