



## 个人资料



Muses\_9

访问：2885次

积分：88

等级：**BLOG > 1**

排名：千里之外

原创：5篇 转载：1篇

译文：0篇 评论：1条

## 文章搜索

## 文章分类

- [C++ \(1\)](#)
- [ros \(1\)](#)
- [c# \(2\)](#)
- [视频图像处理 \(2\)](#)
- [机器人操作系统 \(1\)](#)
- [ffmpeg \(1\)](#)

## 文章存档

- [2017年01月 \(1\)](#)
- [2016年11月 \(2\)](#)
- [2016年08月 \(3\)](#)

## 阅读排行

- [k-means算法实现图像颜色聚类 \(1561\)](#)
- [ffmpeg获取运动矢量及显示 \(402\)](#)
- [初识ROS机器人操作系统 \(347\)](#)
- [c++文件io操作练习之写文件 \(336\)](#)
- [c# 委托与异步调用 \(132\)](#)
- [c#调用DLL \(67\)](#)

## 评论排行

- [ffmpeg获取运动矢量及显示 \(1\)](#)
- [c++文件io操作练习之写文件 \(0\)](#)
- [初识ROS机器人操作系统 \(0\)](#)
- [c# 委托与异步调用 \(0\)](#)
- [k-means算法实现图像颜色聚类 \(0\)](#)

异步赠书：9月重磅新书升级，本本经典 [SDCC 2017之区块链技术实战线上峰会](#) [程序员9月书单](#) [每周荐书：ES6、虚拟现实、物联网（评论送书）](#)

## k-means算法实现图像颜色聚类

标签：[算法](#) [图像处理](#)

2016-11-10 18:41

1564人阅读

[评论\(0\)](#)

[收藏](#)

[举报](#)

分类：

[视频图像处理 \(1\)](#)

版权声明：本文为博主原创文章，未经博主允许不得转载。

k-means算法原理比较简单，有关介绍参考<http://blog.csdn.net/cai0538/article/details/7061922>。原始算法最大的缺陷就是初始聚类中心的选取对效果的影响很大，对此有很多研究者提出了解决方法，网上一搜有大量论文。其中k-means++算是比较简单效果较好的了，参考[http://blog.csdn.net/loadstar\\_kun/article/details/39450615](http://blog.csdn.net/loadstar_kun/article/details/39450615)，<http://blog.csdn.net/ihsin/article/details/41809633>。这里实现了对彩色图像的聚类，效果和opencv的KMeans2方法没差别，只是速度很慢。。

```
1 #include<stdio.h>
2 #include <cstdlib>
3 #include<string>
4 #include<math.h>
5 #include<stdlib.h>
6 #include<vector>
7 #include<string.h>
8 #include<iostream>
9
10 #include <cv.h>
11 #include <highgui.h>
12 using namespace cv;
13 using namespace std;
14
15 struct Tuple{
16     int x;
17     int y;
18     unsigned char b;
19     unsigned char g;
20     unsigned char r;
21 };
22
23
24 const int K=4;//簇的数目
25
26 vector<Tuple> node;//存储原始数据
27 Tuple means[K]; //存储均值中心
28 vector<Tuple> clusters[K]; //存储k个聚类
29 int number;//输入数据的数目
30 int dimension;//输入数据的维数
31
32 //计算欧式距离
33 double getDistXY(Tuple t1,Tuple t2)
34 {
35     return sqrt((float)((t1.b-t2.b)*(t1.b-t2.b)+(t1.g-t2.g)*(t1.g-t2.g)+(t1.r-t2.
36 }
37
38 //根据质心,决定当前元组属于哪个簇
39 int clusterOfTuple(Tuple means[],Tuple tuple)
40 {
41     float dist=getDistXY(means[0],tuple);
42     float dist2=getDistXY(means[1],tuple);
43     float dist3=getDistXY(means[2],tuple);
44     float dist4=getDistXY(means[3],tuple);
45     if(dist<dist2)
46     {
47         if(dist<dist3)
48         {
49             if(dist<dist4)
50                 return 0;
51             else
52                 return 3;
53         }
54         else
55             return 2;
56     }
57     else
58     {
59         if(dist2<dist3)
60         {
61             if(dist2<dist4)
62                 return 1;
63             else
64                 return 3;
65         }
66         else
67             return 2;
68     }
69 }
```

## 推荐文章

- \* CSDN日报20170828——《4个方法快速打造你的阅读清单》
- \* Android检查更新下载安装
- \* 动手打造史上最简单的Recycleview 侧滑菜单
- \* TCP网络通讯如何解决分包粘包问题
- \* SDCC 2017之区块链技术实战线上峰会
- \* 快速集成一个视频直播功能

## 最新评论

- ffmpeg获取运动矢量及显示  
gongchen55983: 您好, 为什么我解析什么视频都是undefined constant or missing &#39...
- Opencv之人脸肤色检测总结  
Muses\_9: 背景太简单 实际应用意义不大

```

43     int label=0;//标示属于哪一簇
44     for(int i=1;i<K;i++)
45     {
46         tmp=getDistXY(means[i],tuple);
47         if(tmp<dist)
48         {
49             dist=tmp;
50             label=i;
51         }
52     }
53     return label;
54 }
55
56 //获得给定簇集的平方误差
57 float getVar(vector<Tuple> clusters[],Tuple means[])
58 {
59     float var=0;
60     for(int i=0;i<K;i++)
61     {
62         vector<Tuple> m=clusters[i];
63         for(int j=0;j<m.size();j++)
64         {
65             var+=getDistXY(m[j],means[i]);
66         }
67     }
68     return var;
69 }
70
71 //获得当前簇的均值(质心)
72 Tuple getMeans(vector<Tuple> cluster)
73 {
74     int num=cluster.size();
75     double meansX=0,meansY=0,meansZ=0;
76     Tuple t;
77     for(int i=0;i<num;i++)
78     {
79         meansX+=cluster[i].b;
80         meansY+=cluster[i].g;
81         meansZ+=cluster[i].r;
82     }
83     t.b=meansX/num;
84     t.g=meansY/num;
85     t.r=meansZ/num;
86     return t;
87 }
88 void ChooseSeeds()//选择k个点作为初始的聚类中心,此处用k-means++算法优化,不是随机选择
89 {
90     number=node.size();
91     srand((unsigned int) time(NULL));
92     int idx=rand()%number;
93     int cnt=0;
94     means[cnt++]=node[idx];//记录选择的均值中心
95     double* dis=(double*)malloc(number*sizeof(double));//记录每个点距离它最近的均值中
96     memset(dis,0x3f,sizeof(dis));
97     while(cnt<K)//求出每个点与距离它最近的均值中心的距离
98     {
99         double sum=0;
100         for(int i=0;i<number;i++)
101         {
102             for(int j=0;j<cnt;j++)
103             {
104                 if(i==j) continue;
105                 dis[i]=min(dis[i],getDistXY(node[i],means[j]));
106             }
107             sum+=dis[i];
108         }
109         for(int i=0;i<number;i++)//归一化,其后可以对应到概率
110         {
111             dis[i]/=sum;
112         }
113         double* cumprobs=(double*)malloc(number*sizeof(double));
114         cumprobs[0]=dis[0];
115         for(int i=1;i<number;i++)

```

```

116     {
117         cumprobs[i]=dis[i]+cumprobs[i-1];
118     }
119     bool* used=(bool*)malloc(number*sizeof(bool));
120     memset(used,true,sizeof(used));
121     used[idx]=false;
122     next:
123     double r= (rand()%1000)*0.001;
124     bool flg=true;
125     for(int i=0;i<number;i++)
126     {
127         if(r<cumprobs[i]&&used[i])//选择满足概率的点作为簇中心
128         {
129             idx=i;
130             flg=false;
131             used[i]=false;
132             break;
133         }
134     }
135     if(flg) goto next; //如果没有找到, 重新产生随机数r继续找
136     means[cnt++]=node[idx];
137     free(cumprobs);
138     free(used);
139 }
140
141 free(dis);
142
143 }
144
145 void KMeans(vector<Tuple> tuples)
146 {
147
148     int i=0;
149     ChooseSeeds();
150     int label=0;
151     //根据默认的质心给簇赋值
152     for(i=0;i!=tuples.size();++i)
153     {
154         label=clusterOfTuple(means,tuples[i]);
155         clusters[label].push_back(tuples[i]);
156     }
157
158     float oldVar=-1;
159     float newVar=getVar(clusters,means);
160     int times=0;
161     while(abs(newVar-oldVar)>=1&&times<100)//当新旧函数值相差不到1即准则函数不发生明
162     {
163         for(i=0;i<K;i++)//更新每个簇的中心点
164         {
165             means[i]=getMeans(clusters[i]);
166
167         }
168         oldVar=newVar;
169         newVar=getVar(clusters,means);//计算新的准则函数值
170         for(i=0;i<K;i++)
171         {
172             clusters[i].clear();
173         }
174         //根据新的质心获得新的簇
175         for(i=0;i!=tuples.size();++i)
176         {
177             label=clusterOfTuple(means,tuples[i]);
178             clusters[label].push_back(tuples[i]);
179         }
180         times++;
181     }
182 }
183
184 }
185
186
187
188

```

```

189 {
190
191     IplImage *img=cvLoadImage("horses2.jpg",-1);
192     int img_w=img->width;
193     int img_h=img->height;
194
195     number=img_w*img_h;
196     dimension=3;
197
198     int i,j;
199     Tuple t;
200     for ( i = 0; i < img_h; i++)
201     {
202         for ( j = 0; j < img_w; j++)
203         {
204             t.x=j;
205             t.y=i;
206             t.b=(unsigned char)*(img->imageData + i*img->widthStep+3*j);
207             t.g=(unsigned char)*(img->imageData + i*img->widthStep+3*j+1);
208             t.r=(unsigned char)*(img->imageData + i*img->widthStep+3*j+2);
209             node.push_back(t);
210         }
211     }
212
213     KMeans(node);
214
215     IplImage *bin=cvCreateImage(cvSize(img->width,img->height),IPL_DEPTH_8U,1);//t
216
217     int val=0;
218     float step=255/(K-1);
219     for(i=0;i<K;i++)
220     {
221         vector<Tuple> m=clusters[i];
222         val=255-i*step;
223         for(j=0;j<m.size();j++)
224         {
225             *(bin->imageData+m[j].y*bin->widthStep+m[j].x)=val;
226         }
227     }
228     cvNamedWindow( "原始图像", 1 );
229     cvShowImage( "原始图像", img );
230
231     cvNamedWindow( "聚类图像", 1 );
232     cvShowImage( "聚类图像", bin );
233     cvSaveImage("horses2_k4_a.jpg",bin);
234     cvWaitKey(0);
235
236     cvDestroyWindow( "原始图像" );
237     cvReleaseImage( &img );
238     cvDestroyWindow( "聚类图像" );
239     cvReleaseImage( &bin );
240
241     return 0;
242 }

```

效果



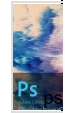
顶 踩  
0 0

上一篇 [ffmpeg获取运动矢量及显示](#)

下一篇 [c#调DLL](#)

#### 相关文章推荐

- [k-means算法实现图像颜色聚类](#)
- [携程机票大数据基础平台架构演进-- 许鹏](#)
- [图片聚类——k-means算法的python实现](#)
- [Python可以这样学--董付国](#)
- [实际项目中以java面向对象的方式实现K-means算...](#)
- [一步一步学Spring Boot](#)
- [层次聚类以及k-means算法](#)
- [深入浅出C++程序设计](#)
- [K-means算法（基于MovieLens数据分别对user和...](#)
- [Android Material Design 新控件](#)
- [K-means算法实战-一维数据的聚类](#)
- [机器学习需要用到的数学知识](#)
- [K-means算法及文本聚类实践](#)
- [随机计算TFIDF作为权重，然后利用余弦距离进行...](#)
- [【python学习笔记】9：用k-means算法对数据进...](#)
- [第24节--聚类之K-means算法（上）](#)



查看评论

暂无评论

发表评论

用户名：  
matrix6666

评论内容：



提交

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

公司简介 | 招贤纳士 | 广告服务 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服   杂志客服   微博客服   webmaster@csdn.net   400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved 