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)
ffmpegg (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 (1561)
ffmpeg获取运动矢量及显 (402)
初识ROS机器人操作系统 (347)
c++文件io操作练习之写[(336)
c# 委托与异步调用 (132)
c#调DLL (67)

评论排行

ffmpeg获取运动矢量及显 (1)
c++文件io操作练习之写[(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 (0)

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）

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/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。。

```
#include<stdio.h>
#include <cstdio>
#include<string>
#include<math.h>
#include<stdlib.h>
#include<vector>
#include<string.h>
#include<iostream>

#include <cv.h>
#include <highgui.h>
using namespace cv;
using namespace std;

struct Tuple{
    int x;
    int y;
    unsigned char b;
    unsigned char g;
    unsigned char r;
};


const int K=4;//簇的数目

vector<Tuple> node;//存储原始数据
Tuple means[K];//存储均值中心
vector<Tuple> clusters[K];//存储k个聚类
int number;//输入数据的数目
int dimension;//输入数据的维数

//计算欧式距离
 double getDistXY(Tuple t1,Tuple t2)
 {
     return sqrt((float)((t1.b-t2.b)*(t1.b-t2.b)+(t1.g-t2.g)*(t1.g-t2.g)+(t1.r-t2.
 }

 //根据质心,决定当前元组属于哪个簇
 int clusterOfTuple(Tuple means[],Tuple tuple)
 {
     float dist=getDistXY(means[0],tuple);
     float tmp;
```

c#调DLL (0)

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++)//求出概率的前缀和
```

```
        {
            cumprobs[i]=dis[i]+cumprobs[i-1];
        }
        bool* used=(bool*)malloc(number*sizeof(bool));
        memset(used,true,sizeof(used));
        used[idx]=false;
        next:
        double r= (rand()%1000)*0.001;
        bool flg=true;
        for(int i=0;i<number;i++)
        {
            if(r<cumprobs[i]&&used[i])//选择满足概率的点作为簇中心
            {
                idx=i;
                flg=false;
                used[i]=false;
                break;
            }
        }
        if(flg) goto next; //如果没有找到，重新产生随机数r继续找
        means[cnt++]=node[idx];
        free(cumprobs);
        free(used);
    }

    free(dis);

}

 void KMeans(vector<Tuple> tuples)
 {

     int i=0;
     ChooseSeeds();
     int label=0;
     //根据默认的质心给簇赋值
     for(i=0;i!=tuples.size();++i)
     {
         label=clusterOfTuple(means,tuples[i]);
         clusters[label].push_back(tuples[i]);
     }

     float oldVar=-1;
     float newVar=getVar(clusters,means);
     int times=0;
     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
     {
         for(i=0;i<K;i++)//更新每个簇的中心点
         {
             means[i]=getMeans(clusters[i]);

         }
         oldVar=newVar;
         newVar=getVar(clusters,means);//计算新的准则函数值
         for(i=0;i<K;i++)
         {
             clusters[i].clear();
         }
         //根据新的质心获得新的簇
         for(i=0;i!=tuples.size();++i)
         {
             label=clusterOfTuple(means,tuples[i]);
             clusters[label].push_back(tuples[i]);
         }
         times++;

     }

 }
```

```
{

    IplImage *img=cvLoadImage("horses2.jpg",-1);
    int img_w=img->width;
    int img_h=img->height;

    number=img_w*img_h;
    dimension=3;

    int i,j;
    Tuple t;
    for ( i = 0; i < img_h; i++)
    {
        for ( j = 0; j < img_w; j++)
        {
            t.x=j;
            t.y=i;
            t.b=(unsigned char)*(img->imageData + i*img->widthStep+3*j);
            t.g=(unsigned char)*(img->imageData + i*img->widthStep+3*j+1);
            t.r=(unsigned char)*(img->imageData + i*img->widthStep+3*j+2);
            node.push_back(t);
        }
    }

    KMeans(node);

    IplImage *bin=cvCreateImage(cvSize(img->width,img->height),IPL_DEPTH_8U,1);//

    int val=0;
    float step=255/(K-1);
    for(i=0;i<K;i++)
    {
        vector<Tuple> m=clusters[i];
        val=255-i*step;
        for(j=0;j<m.size();j++)
        {
            *(bin->imageData+m[j].y*bin->widthStep+m[j].x)=val;
        }
    }
    cvNamedWindow( "原始图像", 1 );
    cvShowImage( "原始图像", img  );

     cvNamedWindow( "聚类图像", 1 );
     cvShowImage( "聚类图像", bin  );
     cvSaveImage("horses2_k4_a.jpg",bin);
     cvWaitKey(0);

     cvDestroyWindow( "原始图像" );
     cvReleaseImage( &img );
     cvDestroyWindow( "聚类图像" );
     cvReleaseImage( &bin );

     return 0;
}
```

效果

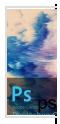
顶 0　　踩 0

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