

zouxy09的专栏

悲喜枯荣如是本无分别，当来则来，当去则去，随心，随性，随缘！ -

目录视图 摘要视图 RSS 订阅

个人资料



zouxy09



访问: 8391822次
积分: 28295
等级:
排名: 第189名
原创: 116篇 转载: 11篇
译文: 1篇 评论: 3828条

个人简介

关注: 机器学习、计算机视觉、人机交互和人工智能等领域。
邮箱: zouxy09@qq.com
微博: Erik-zou
交流请发邮件, 不怎么看博客私信^_^

相关资料与课程推荐

7月在线
JULY 2017

谷歌工程师带路
免费GPU畅爽实战

——《TensorFlow框架案例实战》

269元秒杀详图

四大特色:

- 1) 完整覆盖Tensorflow及上层库
- 2) 从数据预处理到模型训练全流程
- 3) 免费提供GPU服务器供实战
- 4) 精讲GAN图像生成最新案例

扫码拼团:

文章搜索

文章分类

OpenCV (29)

光流Optical Flow介绍与OpenCV实现

2013-03-17 15:53 81983人阅读 评论(22) 收藏 举报

分类: OpenCV (28) 计算机视觉 (72) 图像处理 (54)

版权声明: 本文为博主原创文章, 未经博主允许不得转载。

光流Optical Flow介绍与OpenCV实现

zouxy09@qq.com

<http://blog.csdn.net/zouxy09>

光流 (optical flow) 是什么呢? 名字很专业, 感觉很陌生, 但本质上, 我们是最熟悉不过的了。因为这种视觉现象我们每天都在经历。从本质上说, 光流就是你在这个运动着的世界里感觉到的明显的视觉运动 (呵呵, 相对论, 没有绝对的静止, 也没有绝对的运动)。例如, 当你坐在火车上, 然后往窗外看。你可以看到树、地面、建筑等等, 他们都在往后退。这个运动就是光流。而且, 我们都会发现, 他们的运动速度居然不一样? 这就给我们提供了一个挺有意思的信息: 通过不同目标的运动速度判断它们与我们的距离。一些比较远的目标, 例如云、山, 它们移动很慢, 感觉就像静止一样。但一些离得比较近的物体, 例如建筑和树, 就比较快的往后退, 然后离我们的距离越近, 它们往后退的速度越快。一些非常近的物体, 例如路面的标记啊, 草地啊等等, 快到好像在我们耳旁发出嗖嗖的声音。

光流除了提供远近外, 还可以提供角度信息。与咱们的眼睛正对着的方向成90度方向运动的物体速度要比其他角度的快, 当小到0度的时候, 也就是物体朝着我们的方向直接撞过来, 我们就是感受不到它的运动 (光流) 了, 看起来好像是静止的。当它离我们越近, 就越来越大 (当然了, 我们平时看到感觉还是有速度的, 因为物体较大, 它的边缘还是和我们人眼具有大于0的角度的)。

呵呵, 说了那么多, 好像还没进入比较官方的, 研究性的定义。那就贴上一个吧。

光流的概念是Gibson在1950年首先提出来的。它是空间运动物体在观察成像平面上的像素运动的瞬时速度, 是利用图像序列中像素在时间域上的变化以及相邻帧之间的相关性来找到上一帧跟当前帧之间存在的对应关系, 从而计算出相邻帧之间物体的运动信息的一种方法。一般而言, 光流是由于场景中前景目标本身的移动、相机的运动, 或者两者的共同运动所产生的。

当人的眼睛观察运动物体时, 物体的景象在人眼的视网膜上形成一系列连续变化的图像, 这一系列连续变化的信息不断“流过”视网膜 (即图像平面), 好像一种光的“流”, 故称之为光流 (optical flow)。光流表达了图像的变化, 由于它包含了目标运动的信息, 因此可被观察者用来确定目标的运动情况。

研究光流场的目的就是为从图片序列中近似得到不能直接得到的运动场。运动场, 其实就是物体在三维真实世界中的运动; 光流场, 是运动场在二维图像平面上 (人的眼睛或者摄像头) 的投影。

那通俗的讲就是通过一个图片序列, 把每张图像中每个像素的运动速度和运动方向找出来就是光流场。那怎么找呢? 咱们直观理解肯定是: 第t帧的时候A点的位置是(x₁, y₁),

机器学习 (46)

计算机视觉 (73)

Deep Learning (18)

语音识别与TTS (13)

图像处理 (55)

Linux (15)

Linux驱动 (4)

嵌入式 (18)

OpenAL (3)

Android (1)

C/C++编程 (18)

摄像头相关 (5)

数学 (5)

Kinect (9)

神经网络 (8)

随谈 (2)

文章存档

2015年10月 (4)

2015年04月 (2)

2014年12月 (1)

2014年08月 (1)

2014年05月 (2)

展开

阅读排行

Deep Learning (深度学 (699920)

Deep Learning (深度学 (479466)

Deep Learning (深度学 (463029)

Deep Learning 论文笔记 (320514)

Deep Learning (深度学 (294886)

机器学习中的范数规则化 (264649)

Deep Learning (深度学 (252783)

Deep Learning (深度学 (238662)

从最大似然到EM算法浅 (234420)

Deep Learning (深度学 (203869)

评论排行

Deep Learning 论文笔记 (299)

从最大似然到EM算法浅 (199)

Deep Learning (深度学 (190)

基于Qt的P2P局域网聊天 (167)

时空上下文视觉跟踪 (S (145)

计算机视觉、机器学习相 (120)

机器学习中的范数规则化 (102)

机器学习算法与Python (102)

Deep Learning (深度学 (91)

Deep Learning 论文笔记 (75)

最新评论

Deep Learning (深度学习) 学 (2017-08-02 10:00)

码灵薯: 写的不错, 以前总是想不通为什么深度学习这么厉害, 读完这个系列对深度学习总算有了一些直观的感受, 在此谢...

那么我们在第t+1帧的时候再找到A点，假如它的位置是(x₂,y₂)，那么我们就可以确定A点的运动了：(u_x, v_y) = (x₂, y₂) - (x₁,y₁)。

那怎么知道第t+1帧的时候A点的位置呢？这就存在很多的光流计算方法了。

1981年，Horn和Schunck创造性地将二维速度场与灰度相联系，引入光流约束方程，得到光流计算的基本**算法**。人们基于不同的理论基础提出各种光流计算方法，算法性能各有不同。Barron等人对多种光流计算技术进行了总结，按照理论基础与数学方法的区别把它们分成四种：**基于梯度的方法、基于匹配的方法、基于能量的方法、基于相位的方法**。近年来**神经动力学方法**也颇受学者重视。

其他的咱们先不说了，回归应用吧（呵呵，太高深了，自己说不下去了）。**opencv**中实现了不少的光流算法。

1) calcOpticalFlowPyrLK

通过金字塔Lucas-Kanade 光流方法计算某些点集的光流（稀疏光流）。理解的话，可以参考这篇论文：“Pyramidal Implementation of the Lucas Kanade Feature TrackerDescription of the algorithm”

2) calcOpticalFlowFarneback

用Gunnar Farneback 的算法计算稠密光流（即图像上所有像素点的光流都计算出来）。它的相关论文是：“Two-Frame Motion Estimation Based on PolynomialExpansion”

3) CalcOpticalFlowBM

通过块匹配的方法来计算光流。

4) CalcOpticalFlowHS

用Horn-Schunck 的算法计算稠密光流。相关论文好像是这篇：“Determining Optical Flow”

5) calcOpticalFlowSF

这一个是2012年欧洲视觉会议的一篇文章的实现：“SimpleFlow: A Non-iterative, Sublinear Optical FlowAlgorithm”，工程网站是：<http://graphics.berkeley.edu/papers/Tao-SAN-2012-05/> 在OpenCV新版本中有引入。

稠密光流需要使用某种插值方法在比较容易跟踪的像素之间进行插值以解决那些运动不明确的像素，所以它的计算开销是相当大的。而对于稀疏光流来说，在他计算时需要在被跟踪之前指定一组点（容易跟踪的点，例如角点），因此在使用LK方法之前我们需要配合使用cvGoodFeatureToTrack()来寻找角点，然后利用金字塔LK光流算法，对运动进行跟踪。但个人感觉，对于少纹理的目标，例如人手，LK稀疏光流就比较容易跟丢。

至于他们的API的使用说明，我们直接参考OpenCV的官方手册就行：

http://www.opencv.org.cn/opencvdoc/2.3.2/html/modules/video/doc/motion_analysis_and_o

IJCV2011有一篇文章，《**A Database and Evaluation Methodology for Optical Flow**》里面对主流的光流算法做了简要的介绍和对不同算法进行了评估。网址是：<http://vision.middlebury.edu/flow/>

感觉这个文章在光流算法的解说上非常好，条例很清晰。想了解光流的，推荐看这篇文章。另外，需要提到的一个问题是，光流场是图片中每个像素都有一个x方向和y方向的位移，所以在上面那些光流计算结束后得到的光流**flow**是个和原来图像大小相等的双通道图像。那怎么可视化呢？这篇文章用的是Munsell颜色系统来显示。

Python机器学习库scikit-learn实! Ashley_JIANG: 不好意思, 是初学者, 这段程序做了什么事情呀?

关于计算机视觉 (随谈) 曹学亮: 持续膜拜中。。。

Deep Learning论文笔记之 (五) weixin_39678327: 请问楼主cnnexample里的train_x,train_y可以理解成图片的feature和lab...

Deep Learning论文笔记之 (五) weixin_39678327: 请问楼主cnnexample里的train_x,train_y可以理解成图片的feature和lab...

生成模型与判别模型 StudyAi_com: 讲的很棒, 例子很生动。。。生成模型研究联合概率, 判别模型研究条件概率!! 欢迎楼主互粉studyai....

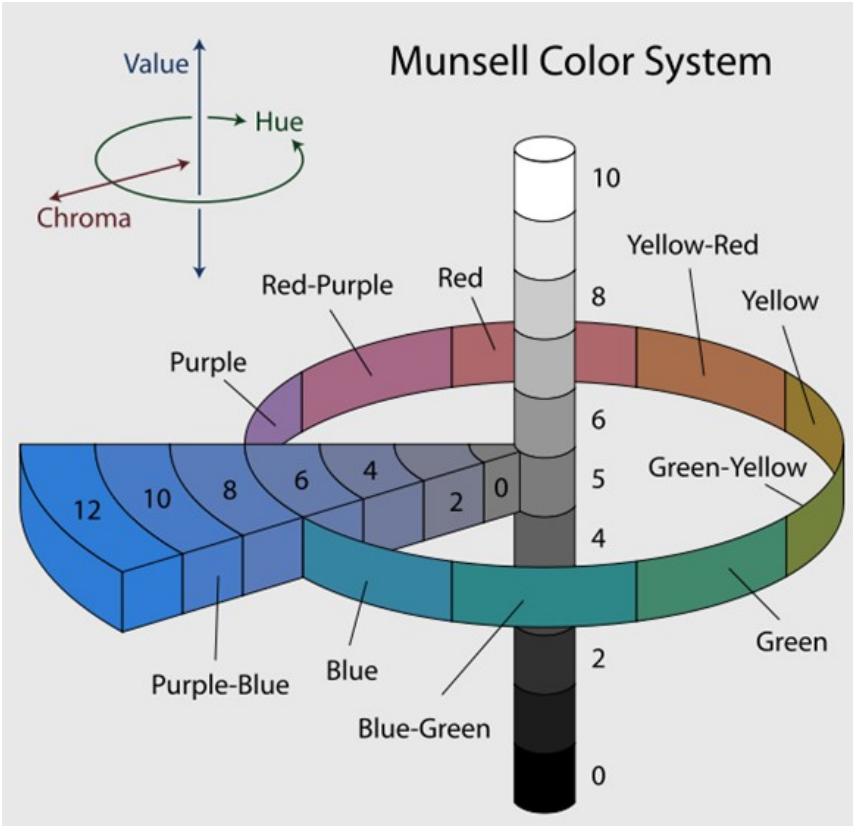
图像分割之 (五) 活动轮廓模型: qq_38276767: 求levelset的讲解

生成模型与判别模型 quankun9683: @whxtbest:你好, 能解释下为什么说生成模型能用于生成数据而判别模型不行吗?

生成模型与判别模型 quankun9683: @SimpleGatsby:你好, 能解释下为什么说生成模型能用于生成数据而判别模型不行吗?

机器学习算法与Python实践之 (迟暮花开: 写的简直太好了, 给博主一万个赞!!!!

关于孟塞尔颜色系统（MunsellColor System），可以看wikibaike。它是美国艺术家阿尔伯特孟塞尔（Albert H. Munsell, 1858—1918）在1898年创制的颜色描述系统。



孟塞尔颜色系统的空间大致成一个圆柱形：

南北轴=明度（value），从全黑（1）到全白（10）。

经度=色相（hue）。把一周均分成五种主色调和五种中间色：红(R)、红黄(YR)、黄(Y)、黄绿(GY)、绿(G)、绿蓝(BG)、蓝(B)、蓝紫(PB)、紫(P)、紫红(RP)。相邻的两个位置之间再均分10份，共100份。

距轴的距离=色度（chroma），表示色调的纯度。其数值从中间（0）向外随着色调的纯度增加，没有理论上的上限（普通的颜色实际上限为10左右，反光、荧光等材料可高达30）。由于人眼对各种颜色的敏感度不同，色度不一定与每个色调和明度组合相匹配。

具体颜色的标识形式为：色相+明度+色度。

在上面的那个评估的网站有这个从flow到color显示的Matlab和C++代码。但是感觉C++代码分几个文件，有点乱，然后我自己整理成两个函数了，并配合OpenCV的Mat格式。

下面的代码是用calcOpticalFlowFarneback来计算稠密光流并且用这个颜色系统来显示的。这个计算稠密光流的方法与其他几个相比还是比较快的，640x480的视频我的是200ms左右一帧，但其他的一般都需要一两秒以上。结果图中，不同颜色表示不同的运动方向，深浅就表示运动的快慢了。

```
void calcOpticalFlowFarneback(InputArray prevImg, InputArray
nextImg,InputOutputArray flow, double pyrScale, int levels, int winsize, int iterations, int
polyN, double polySigma, int flags)
```

大部分参数在论文中都有一套比较好的值的，直接采用他们的就好了。

```
[cpp]
01. // Farneback dense optical flow calculate and show in Munsell system of colors
02. // Author : Zouxy
03. // Date : 2013-3-15
04. // HomePage : http://blog.csdn.net/zouxy09
05. // Email : zouxy09@qq.com
```

```

06.
07. // API calcOpticalFlowFarneback() comes from OpenCV, and this
08. // 2D dense optical flow algorithm from the following paper:
09. // Gunnar Farneback, "Two-Frame Motion Estimation Based on Polynomial Expansion".
10. // And the OpenCV source code locate in ..\opencv2.4.3\modules\video\src\optflowgf.cpp
11.
12. #include <iostream>
13. #include "opencv2/opencv.hpp"
14.
15. using namespace cv;
16. using namespace std;
17.
18. #define UNKNOWN_FLOW_THRESH 1e9
19.
20. // Color encoding of flow vectors from:
21. // http://members.shaw.ca/quadibloc/other/colint.htm
22. // This code is modified from:
23. // http://vision.middlebury.edu/flow/data/
24. void makecolorwheel(vector<Scalar> &colorwheel)
25. {
26.     int RY = 15;
27.     int YG = 6;
28.     int GC = 4;
29.     int CB = 11;
30.     int BM = 13;
31.     int MR = 6;
32.
33.     int i;
34.
35.     for (i = 0; i < RY; i++) colorwheel.push_back(Scalar(255, 255*i/RY, 0);
36.     for (i = 0; i < YG; i++) colorwheel.push_back(Scalar(255-255*i/YG, 255, 0));
37.     for (i = 0; i < GC; i++) colorwheel.push_back(Scalar(0, 255, 255*i/GC));
38.     for (i = 0; i < CB; i++) colorwheel.push_back(Scalar(0, 255-255*i/CB, 255));
39.     for (i = 0; i < BM; i++) colorwheel.push_back(Scalar(255*i/BM, 0, 255));
40.     for (i = 0; i < MR; i++) colorwheel.push_back(Scalar(255, 0, 255-255*i/MR));
41. }
42.
43. void motionToColor(Mat flow, Mat &color)
44. {
45.     if (color.empty())
46.         color.create(flow.rows, flow.cols, CV_8UC3);
47.
48.     static vector<Scalar> colorwheel; //Scalar r,g,b
49.     if (colorwheel.empty())
50.         makecolorwheel(colorwheel);
51.
52.     // determine motion range:
53.     float maxrad = -1;
54.
55.     // Find max flow to normalize fx and fy
56.     for (int i= 0; i < flow.rows; ++i)
57.     {
58.         for (int j = 0; j < flow.cols; ++j)
59.         {
60.             Vec2f flow_at_point = flow.at<Vec2f>(i, j);
61.             float fx = flow_at_point[0];
62.             float fy = flow_at_point[1];
63.             if ((fabs(fx) > UNKNOWN_FLOW_THRESH) || (fabs(fy) > UNKNOWN_FLOW_THRESH))
64.                 continue;
65.             float rad = sqrt(fx * fx + fy * fy);
66.             maxrad = maxrad > rad ? maxrad : rad;
67.         }
68.     }
69.
70.     for (int i= 0; i < flow.rows; ++i)
71.     {
72.         for (int j = 0; j < flow.cols; ++j)
73.         {
74.             uchar *data = color.data + color.step[0] * i + color.step[1] * j;
75.             Vec2f flow_at_point = flow.at<Vec2f>(i, j);
76.
77.             float fx = flow_at_point[0] / maxrad;
78.             float fy = flow_at_point[1] / maxrad;
79.             if ((fabs(fx) > UNKNOWN_FLOW_THRESH) || (fabs(fy) > UNKNOWN_FLOW_THRESH))
80.             {
81.                 data[0] = data[1] = data[2] = 0;
82.                 continue;
83.             }
84.             float rad = sqrt(fx * fx + fy * fy);

```

```

85.
86.         float angle = atan2(-fy, -fx) / CV_PI;
87.         float fk = (angle + 1.0) / 2.0 * (colorwheel.size()-1);
88.         int k0 = (int)fk;
89.         int k1 = (k0 + 1) % colorwheel.size();
90.         float f = fk - k0;
91.         //f = 0; // uncomment to see original color wheel
92.
93.         for (int b = 0; b < 3; b++)
94.         {
95.             float col0 = colorwheel[k0][b] / 255.0;
96.             float col1 = colorwheel[k1][b] / 255.0;
97.             float col = (1 - f) * col0 + f * col1;
98.             if (rad <= 1)
99.                 col = 1 - rad * (1 - col); // increase saturation with radius
100.            else
101.                col *= .75; // out of range
102.            data[2 - b] = (int)(255.0 * col);
103.        }
104.    }
105. }
106. }
107.
108. int main(int, char**)
109. {
110.     VideoCapture cap;
111.     cap.open(0);
112.     //cap.open("test_02.wmv");
113.
114.     if( !cap.isOpened() )
115.         return -1;
116.
117.     Mat prevgray, gray, flow, cflow, frame;
118.     namedWindow("flow", 1);
119.
120.     Mat motion2color;
121.
122.     for(;;)
123.     {
124.         double t = (double)cvGetTickCount();
125.
126.         cap >> frame;
127.         cvtColor(frame, gray, CV_BGR2GRAY);
128.         imshow("original", frame);
129.
130.         if( prevgray.data )
131.         {
132.             calcOpticalFlowFarneback(prevgray, gray, flow, 0.5, 3, 15, 3, 5, 1.2, 0);
133.             motionToColor(flow, motion2color);
134.             imshow("flow", motion2color);
135.         }
136.         if(waitKey(10)>=0)
137.             break;
138.         std::swap(prevgray, gray);
139.
140.         t = (double)cvGetTickCount() - t;
141.         cout << "cost time: " << t / ((double)cvGetTickFrequency()*1000.) << endl;
142.     }
143.     return 0;
144. }

```

这个是效果：

一个挥动的手：



虽然也有背景在动，但是因为他们的运动方向不一样，所以还是可以辨认出其中哪个是个是手，在前景和背景运动不统一的时候，还是可以辨认出来的。



顶 踩

46 2

上一篇 计算机视觉、机器学习相关领域论文和源代码大集合--持续更新.....

下一篇 Android学习笔记之（一）开发环境搭建

相关文章推荐


- 光流法简单介绍
 - 关于OpenCV的那些事——Orb角点检测，BF匹配...
 - 光流金字塔calcOpticalFlowPyrLK
 - 光流法
 - OpenCV 使用光流法检测物体运动
- Opencv学习笔记（九）光流法
 - opencv 稀疏光流 稠密光流
 - OpenCV目标跟踪（二）-LK光流法
 - Opencv3.1.0+opencv_contrib配置及使用SIFT测试
 - OpenCV3.1 xfeatures2d::SIFT 使用

猜你在找

- 【直播】机器学习&深度学习系统实战（唐宇迪）
 - 【直播回放】深度学习基础与TensorFlow实践（王琛）
 - 【直播】机器学习之凸优化（马博士）
 - 【直播】机器学习之概率与统计推断（冒教授）
 - 【直播】TensorFlow实战进阶（智亮）
- 【直播】Kaggle 神器：XGBoost 从基础到实战（冒教授）
 - 【直播】计算机视觉原理及实战（屈教授）
 - 【直播】机器学习之矩阵（黄博士）
 - 【直播】机器学习之数学基础
 - 【直播】深度学习30天系统实训（唐宇迪）


查看评论

18楼 [leeRR5](#) 2017-07-15 14:24发表




在节能灯下,没有物体运动的时候还是会有轮廓? bug ?

17楼 [cherishy_](#) 2016-08-14 12:13发表



谢楼主，最近在学习光流，有源码真是太方便了

16楼 [haithink](#) 2016-04-11 19:44发表




"但个人感觉，对于少纹理的目标，例如人手，LK稀疏光流就比较容易跟丢。" 请问楼主知道现在有哪些方法比较好不? 谢谢！

15楼 [仗笔天涯小小鸟](#) 2015-11-22 21:20发表




赞一个

14楼 [宋易雪](#) 2015-11-21 15:15发表




写的非常好！

13楼 [deng_mark](#) 2015-04-11 22:45发表




请问楼主，利用calcOpticalFlowPyrLK怎么可以计算出光流矢量啊？现在只能进行特征点的匹配跟踪。

12楼 [wenjun2012](#) 2015-01-15 11:28发表




楼主整理的知识条理很清楚，看过后对光溜有了大致的了解

11楼 [tracyliang223](#) 2014-12-17 10:02发表




总结的很好，大的方向了解了光流法。

10楼 [bg2bkk](#) 2014-06-18 13:31发表




谢谢你哦，我很快就找到这个测试集，但是不知道它是干嘛使的，谢谢你。

9楼 [欢乐的工科小硕](#) 2014-01-10 19:18发表




分析的很好，受益。

8楼 [JonathanSong](#) 2013-12-13 14:43发表



感谢楼主写出这么好的博文，受益匪浅。
有个问题想请教楼主，孟塞尔颜色空间可视化光流场，那些颜色是如何代表光流的方向的？代码中如何将不同的方向赋予不同的颜色请楼主解释下，谢谢了！

7楼 [updog](#) 2013-09-05 10:40发表



学习光流很好的帖子！

6楼 [ldpxxx](#) 2013-08-14 13:18发表



这篇文章写得太好了，

5楼 [nank043](#) 2013-07-10 15:02发表



请问lz，calcOpticalFlowFarneback方法是如何计算出光流的？匹配结束的条件是什么啊？

4楼 [yaoqinru](#) 2013-06-03 16:06发表



请问楼主 calcOpticalFlowSF方法在opencv哪个版本有。。谢了

Re: [zouxy09](#) 2013-06-03 23:32发表



回复yaoqinru：在opencv2.4.2版本以上应该都有，demo在
OpenCV\opencv\samples\cpp\simpleflow_demo.cpp
源码在：
OpenCV\opencv\modules\video\src\simpleflow.cpp

Re: [zouxy09](#) 2013-06-03 23:32发表



回复zouxy09：不过这个在我电脑上测试好慢的

3楼 [xiaobulazispf](#) 2013-05-31 18:17发表



很喜欢博主的文章，刚刚用豆约翰博客备份专家备份了您的全部博文。

2楼 [wsc36305](#) 2013-05-30 15:30发表



calcOpticalFlowSF这个貌似很慢啊

Re: [zouxy09](#) 2013-05-31 00:04发表



回复wsc36305：恩恩，是啊，稠密光流的计算普遍比较慢，calcOpticalFlowFarneback还快点，我电脑也就是
200ms，640x480的图像

Re: [wsc36305](#) 2013-05-31 10:18发表



回复zouxy09：我最后选择的也是calcOpticalFlowFarneback，如果显卡比较给力，GPU版的会快很多

1楼 [木笔](#) 2013-04-07 14:42发表



您好吗，程序没有看懂，可以大概解释一下嘛？主要是孟塞尔颜色空间的那个。您的意思是建立一个颜色轮子模拟孟塞尔空间？但是还是没有懂怎么显示的。
另外再问下啊，我想转换到孟塞尔颜色空间求颜色相似性，有没有办法实现RGB到孟塞尔空间的转换啊？谢谢指教

发表评论

用户名: [yangwangsou2971](#)

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