# OpenCV中利用色度直方图反投影实现脸部跟踪

## 一、原理

在不同场景中，脸部的色调比较稳定，因此，考虑在色调图像中寻找脸。直方图反投影图像是一个像素归属概率的图像，它可由样本直方图和一幅待处理图像计算得到，实质是在待处理图像中，寻找颜色特征归属样本的概率。在这里，我们对图像一设置一个ROI区域，其位置是在脸部，计算ROI区域的色度直方图，并以该直方图为参数，在图像二的色调图像中计算反投影图像，得到一张概率图像，归属于脸部的区域，其灰度值较大，而非脸部区域其值较小，但由于除脸部以外，其它物体颜色也可能与脸部相近，但位置应该会与原脸位置有明显区别。为了跟踪脸部的运动，使用meanShift算法对其进行跟踪。

由于饱和度低会使色调信息不可靠，因此在程序中使用一个掩码来屏蔽掉饱和度较低的区域。

## 二、步骤

1、读入图像1

2、设置ROI区域，计算其色调直方图

3、载入图像2

4、分离其HSV通道，在色度通道中计算反投影图像

5、使用meanShift算法进行跟踪

## 三、代码：

#include "stdafx.h"

#include <opencv2\opencv.hpp>

#include "Histogram1D.h"

#include "ColorHistogram.h"

#include "ContentFinder.h"

#include <iostream>

#include <stdlib.h>

using namespace std;

using namespace cv;

CHistogram1D hist1D;

Mat histImg;

int \_tmain(int argc, \_TCHAR\* argv[])

{

//1.读取第一幅狒狒照片

Mat ffImage1 = imread("baboon1.JPG");

if (!ffImage1.data)

{

cout << "can not open picture" << endl;

return 1;

}

//转换到HSV空间

Mat hsvffImage1;

cvtColor(ffImage1, hsvffImage1, COLOR\_RGB2HSV);

//分割通道

vector<Mat>VMat;

split(hsvffImage1, VMat);

//2.设置ROI区域，计算其色度归一化直方图

//设置ROI区域

Rect roiRect(110,260, 30, 30);

Mat roiff1img = VMat[0](roiRect);

//创建roi掩码

Mat roiMask=VMat[1](roiRect);

threshold(roiMask, roiMask, 65, 255, THRESH\_BINARY);

//获取色度直方图

int ch = 0;

int histSize = 256;

float hranges[2];

hranges[0] = 0.0;

hranges[1] = 180.0;

const float\* ranges[1];

ranges[0] = hranges;

MatND hist;

calcHist(&roiff1img, 1, &ch, roiMask, hist, 1, &histSize, ranges);

normalize(hist, hist, 1.0);

//3.载入第二幅图像

Mat ffImage2 = imread("baboon3.jpg");

if (!ffImage2.data)

{

cout << "can not open picture" << endl;

return 1;

}

//转换到HSV空间

Mat hsvffImage2;

cvtColor(ffImage2, hsvffImage2, COLOR\_RGB2HSV);

//分割通道

vector<Mat>VMat2;

split(hsvffImage2, VMat2);

//4、计算第二幅图像的反投影图像

Mat result;

calcBackProject(&VMat2[0], 1,&ch, hist, result, ranges, 255);

//创建掩码

Mat mask = VMat2[1];

threshold(mask, mask, 65, 255, THRESH\_BINARY);

//去除饱和度较低的部分

bitwise\_and(result, mask, result);

//5、meanshift算法跟踪

TermCriteria criteria(TermCriteria::MAX\_ITER, 10, 0.01);

Rect newRect = roiRect;

meanShift(result, newRect, criteria);

//6显示结果

rectangle(ffImage1, roiRect,Scalar(0,0,255));

rectangle(ffImage2, newRect, Scalar(0, 255, 255));

imshow("原图像1", ffImage1);

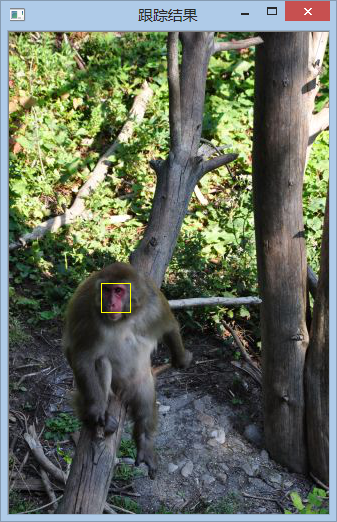
imshow("跟踪结果", ffImage2);

waitKey(0);

return 0;

}

## 四、结果：

## 五、注意

在OpenCV中，计算反投影前需要对直方图进行归一化