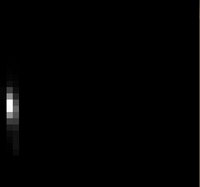
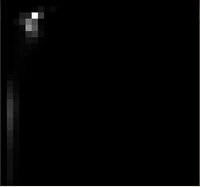
1.概述

反向投影是一种记录给定图像中的像素点如何适应直方图模型像素分布的方式，简单来讲，反向投影就是首先计算某一特征的直方图模型，然后使用模型去寻找图像中存在的特征。反向投影在某一位置的值就是原图对应位置像素值在原图像中的总数目。

2.反向投影原理

原理采用OpenCV docs介绍！使用肤色直方图来解释反向投影的工作原理。假设我们已经获得一个肤色直方图(Hue-Staturation),旁边的直方图就是模型直方图(代表手掌的肤色色调)，可以通过掩码操作来抓取手掌所在区域的直方图：   
   


我们需要做的就是使用模型直方图（代表手掌的皮肤色调）来检测测试图像中的皮肤区域。以下是检测步骤：   
**a.**对测试图像中的每个像素 p(i,j),获取色调数据并找到该色调这里写图片描述在直方图中的bin位置   
**b.**查询模型直方图中对应的bin-这里写图片描述并读取该bin的数值。   
**c.**将此数值存储在新的图像中(BackProjection)。也可以先归一化模型直方图，这样测试图像的输出就可以在屏幕上显示了。   
**d.**通过对测试中的图像中的每个像素采用以上步骤，可以得到如下的BackProjection结果图：   
   
**e.**使用统计学的语言，BackProjection中存储的数值代表了测试图像中该像素属于皮肤区域的概率。以上图为例，亮的区域是皮肤区域的可能性更大，而暗的区域则表示更低的可能性。

这个过程可以笼统的说与计算图像直方图相反，由图像计算直方图的过程比较容易理解，就是统计图像中像素分布的概率，而反向投影正好反过来，是通过直方图来形成图像，其步骤有点类似于直方图均衡化，只不过直方图均衡化是将图像中的每个像素值由一个地方迁移到另外一个地方，而反向投影是直接去直方图中的值，如某种像素值在直方图中的值越大，在进行反向投影操作时其对应的像素值越大即月亮，反过来，如果某灰度值所占面积越小，其反向投影后像素值就会更小。

举个简单的例子来帮助理解这段话的意思。例如灰度图像的像素值如下如下   
grayImage

0 1 2 3   
4 5 6 7   
8 9 10 11   
8 9 14 15   
对图像进行直方图统计（bin指定的区间为[0,3)，[4,7)，[8,11)，[12,16)）如下所示：   
Histogram=

4 4 6 2   
也就是说在[0,3)这个区间的像素值有4个，其它含义相同   
根据上述的直方图进行反向投影，得到反向投影图像像素值如下：   
Back\_Projection=

4 4 4 4   
4 4 4 4   
6 6 6 6   
6 6 2 2   
例如位置(0,0)上的像素值为0，对应的bin为[0,3)，所以反向直方图在该位置上的值这个bin的值4，而在位置(3,3)上的像素为15，其在直方图中的统计为2，故其反向投影图像中的像素为2

3.OpenCV提供API

**calcBackProjection()**函数共有三种形式，根据传入参数的不同选择不同的调用，为重载函数

void cv::calcBackProject ( const Mat \* images,

int nimages,

const int \* channels,

InputArray hist,

OutputArray backProject,

const float \*\* ranges,

double scale = 1,

bool uniform = true

)

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9

参数解释：   
**const Mat\* images**:输入图像，图像深度必须位CV\_8U,CV\_16U或CV\_32F中的一种，尺寸相同，每一幅图像都可以有任意的通道数   
**int nimages**:输入图像的数量   
**const int\* channels**:用于计算反向投影的通道列表，通道数必须与直方图维度相匹配，第一个数组的通道是从0到image[0].channels()-1,第二个数组通道从图像image[0].channels()到image[0].channels()+image[1].channels()-1计数   
**InputArray hist**:输入的直方图，直方图的bin可以是密集(dense)或稀疏(sparse)   
**OutputArray backProject**:目标反向投影输出图像，是一个单通道图像，与原图像有相同的尺寸和深度   
**const float** ranges\*\*:直方图中每个维度bin的取值范围   
**double scale=1**:可选输出反向投影的比例因子   
**bool uniform=true**:直方图是否均匀分布(uniform)的标识符，有默认值true

**另外两种定义形式如下：**

void cv::calcBackProject ( const Mat \* images,

int nimages,

const int \* channels,

const SparseMat & hist,

OutputArray backProject,

const float \*\* ranges,

double scale = 1,

bool uniform = true

)

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9

void cv::calcBackProject ( InputArrayOfArrays images,

const std::vector< int > & channels,

InputArray hist,

OutputArray dst,

const std::vector< float > & ranges,

double scale

)

* 1
* 2
* 3
* 4
* 5
* 6
* 7

4.示例代码

#include <iostream>

#include <opencv2/core.hpp>

#include <opencv2/highgui.hpp>

#include <opencv2/imgproc.hpp>

using namespace std;

using namespace cv;

//定义全局变量

Mat srcImage, hsvImage,hueImage;

const int hueBinMaxValue = 180;

int hueBinValue=25;

//声明回调函数

void Hist\_and\_Backprojection(int, void\*);

int main()

{

srcImage=imread("Back\_Projection\_Theory2.jpg");

//判断图像是否加载成功

if(srcImage.empty())

{

cout << "图像加载失败" << endl;

return -1;

}

else

cout << "图像加载成功..." << endl << endl;

//将图像转化为HSV图像

cvtColor(srcImage, hsvImage, CV\_BGR2HSV);

//只使用图像的H参数

hueImage.create(hsvImage.size(), hsvImage.depth());

int ch[]={0,0};

mixChannels(&hsvImage, 1, &hueImage, 1, ch, 1);

//轨迹条参数设置

char trackBarName[20];

sprintf(trackBarName,"Hue bin:%d",hueBinMaxValue);

namedWindow("SourceImage",WINDOW\_AUTOSIZE);

//创建轨迹条并调用回调函数

createTrackbar(trackBarName, "SourceImage", &hueBinValue, hueBinMaxValue, Hist\_and\_Backprojection);

Hist\_and\_Backprojection(hueBinValue, 0);

imshow("SourceImage", srcImage);

waitKey(0);

return 0;

}

void Hist\_and\_Backprojection(int, void\*)

{

MatND hist;

int histsize=MAX(hueBinValue, 2);

float hue\_range[]={0,180};

const float\* ranges={hue\_range};

//计算图像直方图并归一化处理

calcHist(&hueImage, 1, 0, Mat(), hist, 1, &histsize, &ranges, true, false);

normalize(hist, hist, 0, 255, NORM\_MINMAX, -1, Mat());

//获取反向投影

MatND backProjection;

calcBackProject(&hueImage, 1, 0, hist, backProjection, &ranges, 1, true);

//输出反向投影

imshow("BackProjection", backProjection);

//绘制图像直方图

int w=400;

int h=400;

int bin\_w = cvRound((double)w/histsize);

Mat histImage = Mat::zeros(w, h, CV\_8UC3);

for(int i=0; i < hueBinValue; i++)

{

rectangle(histImage, Point(i\*bin\_w, h), Point((i+1)\*bin\_w, h-cvRound(hist.at<float>(i)\*h/255.0)), Scalar(0,0,255), -1);

}

imshow("HistImage", histImage);

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56
* 57
* 58
* 59
* 60
* 61
* 62
* 63
* 64
* 65
* 66
* 67
* 68
* 69
* 70
* 71
* 72
* 73
* 74
* 75
* 76
* 77
* 78
* 79
* 80
* 81
* 82

程序运行结果：   
