彩色圖像高斯反向投影

lovelygirl OpenCV學堂 昨天

彩色圖像高斯反向投影

一：介紹

圖像反向投影的最終目的是獲取ROI然後實現對ROI區域的標註、識別、測量等圖像處理與分析，是計算機視覺與人工智能的常見方法之一。圖像反向投影通常是彩色圖像投影效果會比灰度圖像效果要好，原因在於彩色圖像帶有更多對象細節信息，在反向投影的時候更加容易判斷、而轉為灰度圖像會導致這些細節信息丟失、從而導致分割失敗。最常見的是基於圖像直方圖特徵的反向投影。我們這裡介紹一種跟直方圖反向投影不一樣的彩色圖像反向投影方法，通過基於高斯的概率分佈公式（PDF）估算，反向投影得到對象區域，該方法也可以看做最簡單的圖像分割方法。缺點是對象顏色光照改變和尺度改變不具備不變性特徵。所以需要在光照度穩定情況下成像採集圖像數據。在這種情況下使用的高斯概率密度公式為：

$$P(r) = \frac{1}{\sigma_r \sqrt{2\pi}} \exp\left\{ -\frac{(r-\mu_r)^2}{2\sigma_r^2} \right\}$$

其中μ表示均值、σ表示标准方差

算法实现步骤为：

1. 輸入模型M，對M的每個像素點（R,G,B）計算I=R+G+B r=R/I, g=G/I, b=B/I
2. 根據得到權重比例值，計算得到對應的均值與標準方差
3. 對輸入圖像的每個像素點計算根據高斯公式計算P(r)與P(g)的乘積
4. 歸一化之後輸出結果，即為最終基於高斯PDF的反向投影圖像

二：算法步驟與代碼實現

1. 首先加載模型圖像與測試圖像
2. 根據模型圖像計算得到每個通道對應的均值與標準方差參數
3. 根據參數方差計算每個像素點的PDF值
4. 歸一化概率分佈圖像-即為反向投影圖像，顯示
5. 根據Mask得到最終顏色模型對象分割

完整的基於OpenCV的C++代碼如下：

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <math.h>

using namespace cv;
using namespace std;

int main(int argc, char** argv) {
    // 加载模型图像与测试图像
```

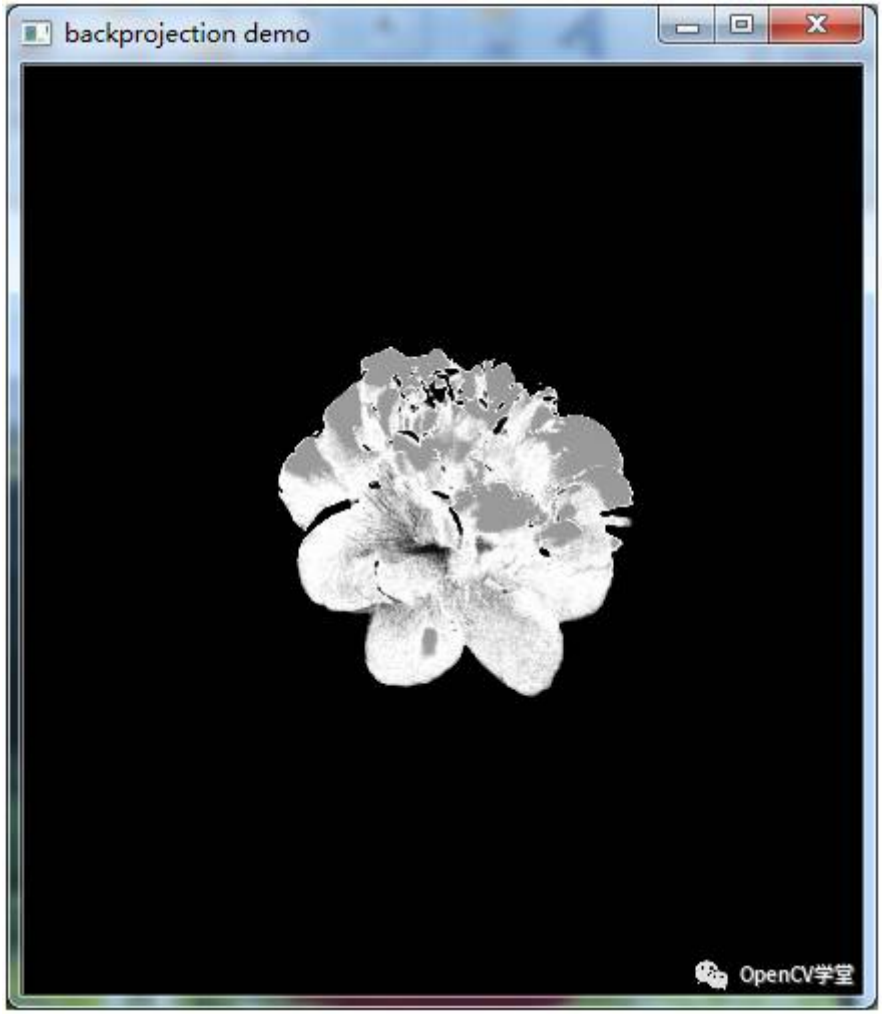
```
    Mat src = imread("D:/gloomyfish/gc_test.png");
    Mat model = imread("D:/gloomyfish/gm.png");
    if (src.empty() || model.empty()) {
        printf("could not load image...\n");
        return -1;
    }
    imshow("input image", src);

    // 对每个通道 计算高斯PDF的参数
    // 有一个通道不计算，是因为它可以通过1-r-g得到
    // 无需再计算
    Mat R = Mat::zeros(model.size(), CV_32FC1);
    Mat G = Mat::zeros(model.size(), CV_32FC1);
    int r = 0, g = 0, b = 0;
    float sum = 0;
    for (int row = 0; row < model.rows; row++) {
        uchar* current = model.ptr<uchar>(row);
        for (int col = 0; col < model.cols; col++) {
            b = *current++;
            g = *current++;
            r = *current++;
            sum = b + g + r;
            R.at<float>(row, col) = r / sum;
            G.at<float>(row, col) = g / sum;
        }
    }

    // 计算均值与标准方差
    Mat mean, stddev;
    double mr, devr;
    double mg, devg;
    meanStdDev(R, mean, stddev);
    mr = mean.at<double>(0, 0);
    devr = mean.at<double>(0, 0);

    meanStdDev(G, mean, stddev);
    mg = mean.at<double>(0, 0);
    devg = mean.at<double>(0, 0);

    int width = src.cols;
    int height = src.rows;

    // 反向投影
    float pr = 0, pg = 0;
    Mat result = Mat::zeros(src.size(), CV_32FC1);
    for (int row = 0; row < height; row++) {
        uchar* currentRow = src.ptr<uchar>(row);
        for (int col = 0; col < width; col++) {
            b = *currentRow++;
            g = *currentRow++;
            r = *currentRow++;
            sum = b + g + r;
            float red = r / sum;
            float green = g / sum;
            pr = (1 / (devr*sqrt(2 * CV_PI)))*exp(-(pow((red - mr), 2)) / (2 * pow(devr, 2)));
            pg = (1 / (devg*sqrt(2 * CV_PI)))*exp(-(pow((green - mg),2)) / (2 * pow(devg, 2)));
            sum = pr*pg;
            result.at<float>(row, col) = sum;
        }
    }
```

```
70.
71.     // 归一化显示高斯反向投影
72.     Mat img(src.size(), CV_8UC1);
73.     normalize(result, result, 0, 255, NORM_MINMAX);
74.     result.convertTo(img, CV_8U);
75.     Mat segmentation;
76.     src.copyTo(segmentation, img);
77.
78.     // 显示
79.     imshow("backprojection demo", img);
80.     imshow("segmentation demo", segmentation);
81.
82.     waitKey(0);
83.     return 0;
84. }
```

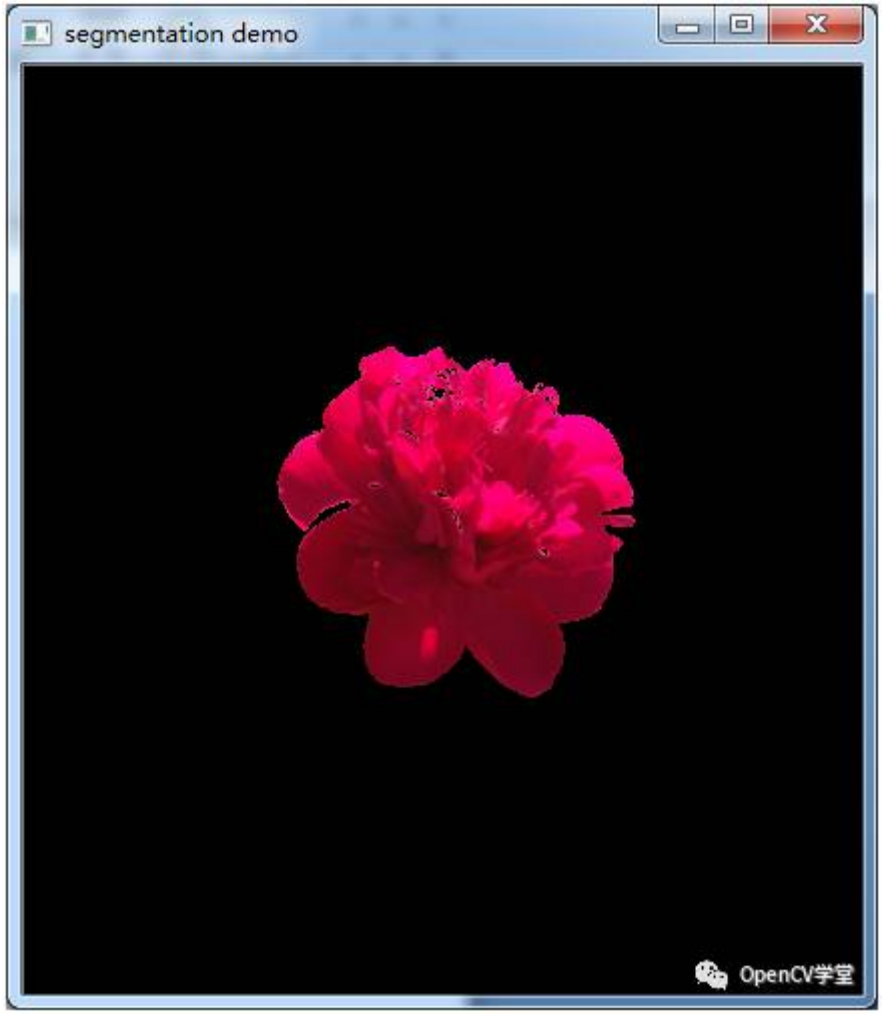
三：測試圖像與效果演示

藍色矩形框為模型，整個圖像為測試圖像

反向投影結果



分割提取結果



四：總結

大家看了這個例子總是有點怪怪的，總會想起點什麼，如果你能想起點什麼的話就是GMM，高斯混合模型，高斯混合模型正是在此基礎上進一步演化而來。

治療對未來焦慮的良藥就在今天你自己的所為