Python手势识别与控制

杨佳锐
2016130078

Abstract

本文中的手势识别与控制功能主要采用*OpenCV*库实现, *OpenCV*是一个基于*BSD*许可(开源)发行的跨平台计算机视觉库, 可以运行在*Linux, Windows, Android*和*Mac-OS*操作系统上. 它轻量级而且高效——由一系列 *C* 函数和少量 *C++* 类构成, 同时提供了*Python, Ruby, MATLAB*等语言的接口, 实现了图像处理和计算机视觉方面的很多通用算法。

主要使用了*OpenCV*的视频采集, 图像色域转换, 颜色通道分割, 高斯滤波, *OSTU*自动阈值, 凸点检测, 边缘检测, 余弦定理计算手势等功能。

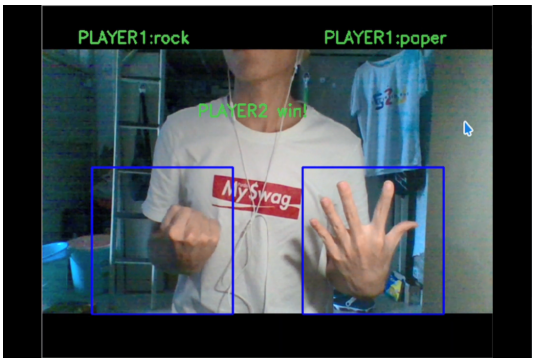

图 1. a demo of this project

1. Introduction

通过Python手势识别与控制，编写一个自动识别手势，进行剪刀石头布的游戏。

1.1. data collection

通过摄像头直接采集数据，后面将直接对数据进行处理。

1.2. data preprocessing

为了更准确地识别时评数据中包含的手势信息，需要对视频数据进行预处理，包括背景减除、人体皮肤侦测。

1.2.1 background subtraction

背景减除我们使用OpenCV中的BackgroundSubtractorMOG2算法，是一个以混合高斯模型为基础的前景/背景分割算法。它是P.KadewTraKuPong和 R.Bowden 在 2001 年提出的。它使用 K（K=3 或 5）个高斯分布混合对背景像素进行建模。使用这些颜色（在整个视频中）存在时间的长短作为混合的权重。背景的颜色一般持续的时间最长，而且更加静止。一个像素怎么会有分布呢？在 x，y 平面上一个像素就是一个像素没有分布，但是我们现在讲的背景建模是基于时间序列的，因此每一个像素点所在的位置在整个时间序列中就会有很多值，从而构成一个分布。

在编写代码时，我们需要使用函数：cv2.createBackgroundSubtractorMOG()创建一个背景对象。这个函数有些可选参数，比如要进行建模场景的时间长度，高斯混合成分的数量，阈值等。将他们全部设置为默认值。然后在整个视频中我们是需要使用backgroundsubtractor.apply() 就可以得到前景的掩模了。

BackgroundSubtractorMOG2 也是以高斯混合模型为基础的背景/前景分割算法。它是以 2004 年和 2006 年 Z.Zivkovic 的两篇文章为基础的。这个算法的一个特点是它为每一个像素选择一个合适数目的高斯分布。（上一个方法中我们使用是 K 高斯分布）。这样就会对由于亮度等发生变化引起的场景变化产生更好的适应。

和前面一样我们需要创建一个背景对象。但在这里我们我们可以选择是否检测阴影。如果 detectShadows = True（默认值），它就会检测并将影子标记出来，但是这样做会降低处理速度。影子会被标记为灰色。

1.2.2 skin detection

皮肤检测采用最大类间方差法 (OTSU [1]),由大津展之于1978年提出的最大类间方差法，是引起较多关注的一种阈值选取方法。它是在判决分析或最小二乘原理的基础上推导出来的。

假设一幅图像有L个灰度级[1,2,⋯,L]。灰度级为i的像素点的个数为ni，那么总的像素点个数就应该为N=n1+n2+⋯+nL。为了讨论方便，我们使用归一化的灰度级直方图并且视为这幅图像的概率分布:

$$p_i = n_i/N, p_i \geq 0, \sum_{i=1}^{L} p_i = 1. \tag{1}$$

现在假设我们通过一个灰度级为k的门限将这些像素点划分为两类:C0和C1(背景和目标,或者反之亦然)；C0表示灰度级为[1,⋯,k]的像素点，C1表示灰度级为[k+1,⋯,L]的像素点。那么，每一类出现的概率以及各类的平均灰度级分别由下面的式子给出：

$$\omega_0 = Pr(C_0) = \sum_{i=1}^{k} p_i = \omega(k) \tag{2}$$

$$\omega_0 = Pr(C_1) = \sum_{i=k+1}^{L} p_i = 1 - \omega(k) \tag{3}$$

以及

$$\mu_0 = \sum_{i=1}^{k} i\, Pr(i|C_0) = \sum_{i=1}^{k} i p_i/\omega_0 = \mu(k)/\omega(k) \tag{4}$$

$$\mu_1 = \sum_{i=k+1}^{L} i\, Pr(i|C_1) = \sum_{i=k+1}^{L} i p_i/\omega_1 = \frac{\mu_T - \mu(k)}{1 - \omega(k)} \tag{5}$$

其中

$$\omega(k) = \sum_{i=1}^{k} p_i \tag{6}$$

$$\mu(k) = \sum_{i=1}^{k} i p_i \tag{7}$$

分别为灰度级从1到k的累积出现概率和平均灰度级（一阶累积矩），而

$$\mu_T = \mu(L) = \sum_{i=1}^{L} i p_i \tag{8}$$

是整幅图像的平均灰度级。我们可以很容易验证，对于任意选定的k，都有：

$$\omega_o \mu_0 + \omega_1 \mu_1 = \mu_T, \qquad \omega_0 + \omega_1 = 1 \tag{9}$$

这两类的类内方差由下面的公式给出：

$$\sigma_0^2 = \sum_{i=1}^{k} (i - \mu_0)^2 Pr(i|C_0) = \sum_{i=1}^{k} (i - \mu_0)^2 p_i/\omega_0 \tag{10}$$

$$\sigma_1^2 = \sum_{i=k+1}^{L} (i - \mu_1)^2 Pr(i|C_1) = \sum_{i=k+1}^{L} (i - \mu_1)^2 p_i/\omega_1 \tag{11}$$

这需要二阶累积矩(second-order cumulative moment，统计学概念)。

为了评价(灰度级k)这个门限“好”的程度，我们需要引入判别式分析中使用的判别式标准来测量（类的分离性测量）：

$$\lambda = \sigma_B^2/\sigma_W^2, \;\; \kappa = \sigma_T^2/\sigma_W^2, \;\; \eta = \sigma_B^2/\sigma_T^2 \tag{12}$$

其中：

$$\sigma_W^2 = \omega_0 \sigma_0^2 + \omega_1 \sigma_1^2 \tag{13}$$

$$\sigma_B^2 = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 = \omega_0 \omega_1 (\mu_1 - \mu_0)^2 \tag{14}$$

又根据式(9)，可以得出：

$$\sigma_T^2 = \sum_{i=1}^{L} (i - \mu_T)^2 p_i \tag{15}$$

这三个式子分别是类内方差、类间方差和灰度级的总方差。然后，我们的问题就简化为一个优化问题，即寻找一个门限κ使(12)式中给出的一个目标函数取最大值。

这个观点是出于这样一个猜想，一个好的阈值将会把灰度级分为两类，那么反过来说，就是如果一个门限能够在灰度级上将图像分割为最好的两类的话，那么

这个门限就是最好的门限。

上面给出的判别式标准是分别求取λ、κ和η的最大值。然而，对于κ而言，它又等于另外一个，比如$\kappa = \lambda+1$；而对于λ而言，又有$\eta = \lambda/(\lambda+1)$，因为始终存在下面的基本关系：

$$\sigma_W^2 + \sigma_B^2 = \sigma_T^2. \tag{16}$$

我们可以发现ω_W^2和ω_B^2都是门限κ的函数，但是ω_T^2却与κ无关。我们还注意到ω_W^2是基于二阶统计（类方差），而ω_B^2是基于一阶统计（类均值）。因此，η是判别κ选取好坏的最简单的测量标准。因此，我们选取η作为评价选择κ作为门限的“好坏”（分离性）的测量标准。

我们使用下面的公式选择不同的κ值顺序搜索，根据式(6)和(7)，或者ji间接使用式(2)到(5)寻找最佳门限κ^*使得η取得最大值，或者等价于使ω_B^2达最大。

$$\eta(k) = \sigma_B^2(k)/\sigma_T^2 \tag{17}$$

$$\sigma_B^2(k) = \frac{[\mu_T\ \omega(k) - \mu(k)]^2}{\omega(k)[1-\omega(k)]} \tag{18}$$

并且，最佳门限κ^*就是

$$\sigma_B^2(\kappa^*) = \max_{1\le k<L} \sigma_B^2(k) \tag{19}$$

1.3. gesture detection and recognition

利用opencv提供的 convexityDefects 凹点检测函数检测图像凹陷的点, 然后利用, 然后根据凹陷点中的 (开始点, 结束点, 远点)的坐标, 利用余弦定理计算两根手指之间的夹角, 其必为锐角, 根据锐角的个数判别手势。其中,锐角个数为0 ,表示手势是拳头或一,锐角个数为0 ,表示手势是拳头或一,锐角个数为1 ,表示手势是剪刀锐角个数为2 ,表示手势是三,锐角个数为3 ,表示手势是四,锐角个数为4 ,表示手势是布。

1.3.1 depression points

对象上的任何凹陷都被成为凸缺陷。OpenCV 中有一个函数 cv.convexityDefect() 可以帮助我们找到凸缺陷。如果要查找凸缺陷，在使用函数 cv2.convexHull 找凸包时，参数returnPoints一定要是 False。最后它会返回一个数组，其中每一行包含的值是：起点，终点，最远的点，到最远点的近似距离。

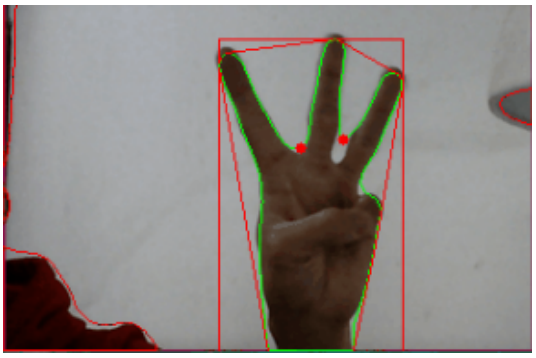

图 2. gesture

1.3.2 gesture control

通过以上方法，就可以实现一个简单的视频识别石头剪刀布的程序，当然，还可以使用以上手势完成各种电脑操作的控制，诸如向上、向下滑动窗口，播放、暂停歌曲等等，都可以通过Python接口实现。

参考文献

[1] N. Otsu. Threshold selection method from gray-level histograms. *IEEE Trans.syst.man Cybern*, 9(1):62–66, 1979.