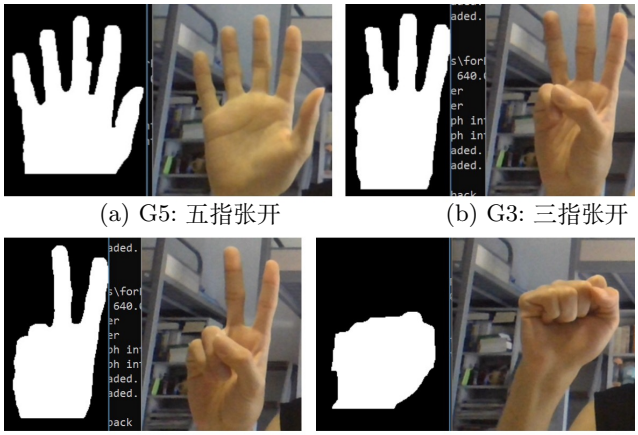
基于图像处理和机器学习的手势识别

钟涛
南开大学

摘要

通过普通摄像头识别镜头前人物的手势，可以作为一种成本低、操作简便的人机交互方式的基础。而如果要做到摄像头前的手势识别，需要完成两个步骤，一是手部的定位，二是手势的判别。本文正是基于此思路设计的解决方案，对于手部定位采用机器学习的方法，通过调用训练好的目标检测之神经网络模型，对手部这一目标进行定位。而对于手势的判别，我在传统数字图像处理的方法基础之上，结合一些几何条件的判断来完成判别。这套解决方案最终能实现流畅地识别镜头前的手势 (FPS>10)，并且识别种类可达四种，每种手势的识别准确率均高于 50%。



(a) G5: 五指张开　(b) G3: 三指张开

(c) G2: 两指张开　(c) G0: 握拳

图 1. 四种常见手势的分割图及其说明。

1. 引言

现在大部分的电脑游戏和软件都要求用户通过鼠标来操控，也有一些游戏公司开发了专门用于游戏的套件，但是无论是鼠标还是专用游戏设备，它们的交互性虽然很强，却都具有不易快速部署、不方便携带的缺点，后者还因价格昂贵而导致其普及率较低。这个时候如果能用普通摄像头完成交互任务，那么不仅可以降低一些游戏及软件的使用门槛，还能减轻用户的经济负担，具有一定实际意义。

手势作为人类的一种肢体语言，能够传达丰富的信息，通过不同手指的组合，就可以形成许多具有很高辨识度的手势，如图1中所示为四种常见的简单手势：五指张开（G5）、三指张开（G3）、两指张开（G2）、握拳（G0）。在现实生活中，它们的具体含义随文化背景的变化而变化，当用于人机交互时，目前也尚无标准指明每个手势的用途及含义，在这里我提出了一个映射规则，为三种手势状态（G5、G3、G0）赋予了特定的应用含义：

- G5：表明释放、取消的状态
- G3：表明移动、拖拽的状态
- G0：表明激活、确认的状态

特别的，由某个手势状态切换至另一状态时，可以触发一种动作，以 G5 和 G0 状态为例，G5→G0 代表由取消态到确认态，相当于鼠标的左键点击动作，我们可以称这样一个状态之间的切换为一级触发器，显然触发器的种类取决于手势状态的数量，不仅如此，多个一级触发器之间的切换也可以构成一个触发器，这时我们称之为二级触发器。可以看到简单的几种手势状态就可以形成足够多的触发器来触发计算机的大部分操作指令，当然触发器的级数越高，也意味着用户操作的难度越大、失败率越高。

既然要用手势识别完成人机交互，那么更重要的是如何通过普通的摄像头识别出镜头前人物的手势，即我们需要设计出一套方法，该方法以普通摄像头实时采集到的视频画面作为输入，以程序运行判断得到的手势类型作为结果输出。由于普通摄像头获取的图像

信息只具备二维信息，不含有物体深度等辅助信息，因此我们在实际设计时需要考虑和解决的问题也比较多。在 Yeo[4]等人提出的一套方法中，采用了这样一个流程来处理摄像头采集的图像：用图像减运算去除背景，用 Haar 特征方法检测人脸并去除，用 Canny 方法检测手部边缘以便提取轮廓，在 YCrCb 颜色空间分割肤色，再用图像形态学运算平滑检测结果并输出。这套方法虽然从视频帧画面中成功提取出了人的手部，但是可以看到，它仅仅依靠了图像处理的手段，而这些步骤中的判断大都是基于一些人类的先验知识来完成的（例如该方法假定视频背景是静止的，假定手部的肤色约束在定值区间内的），因此该方法对环境的适应性较差，容易受到外界因素的干扰。

我在考虑了以上问题后，采用了机器学习的方法来检测出手部的大概位置，从而缩小了手势识别的范围，减小了待处理区域的面积，降低任务的复杂度。这种方法依靠神经网络强大的特征提取能力，可以有效地规避人脸与人的手部肤色相近而引发的误识别问题（如图2），同时也尽可能地排除掉了周围环境潜在的干扰因素，因此具有较强的适应性。

(a) Yeo 的方法　　(b) 我的方法

图 2. Yeo[4]的方法和我的方法对比。视频帧背景进行了全黑填充的处理，而手部进行了白色分割，可以看到我的方法没有把人脸考虑进去。

同时，受限于数据集的缺少和个人计算机算力的不足，我没有直接用神经网络输出具体的手势类别，而是仅仅检测出了手部的位置，在这之后仍然需要进行手势判别的操作。手势判别的过程与 Yeo[4]的一些步骤类似，也用到了 YCrCb 颜色空间分割，形态学运算，轮廓查找等方法。其中对于 G0 手势1，因其与 G5、G3、G2 手势的矩形框包括范围存在差异，我也做了一定的调整以提高手势判别的成功率。

最终，我对图1三种手势进行了测试，用每 60 帧中正确识别出手势的帧数的比例（AR）来衡量手势识别的精度，最终得到 G0、G2、G3、G5 三种手势的识别精度均在 50% 以上。

2. 相关工作

正如前文叙述的，我将手势识别分解为手部定位和手势判别这两个步骤，先进行手部定位这个过程是为了获取一个以人手为中心的感兴趣区域，这样可以排除掉一些全局性信息的干扰，同时也为后一步工作的处理减轻负担。而之后的手势定位采用传统图像处理的方法，深入分析图像的轮廓信息，由此判断出手势所属的类别。

手部定位采用目标检测领域所使用的比较成熟的方法，用 SSD 结构的卷积神经网络实现对“手”这一目标的检测，从而获取其位置信息，由于目标检测领域通常的做法是返回一个包含检测结果的矩形框，所以我的程序中也将这一矩形框信息作为下一步手势判别的输入。另外，由于计算设备的原因，我没有从头训练神经网络，而是直接从 Dibia[3]的项目中获取了训练好的神经网络模型，该模型的实际使用效果符合摄像头前手势识别这一场景的要求。

手势识别这一环节体现了我的方法的独特性，虽然采用的是传统的图像处理方法，但是与 Yeo[4]提出的方法不同之处在于，我所需处理的区域不包含人脸，省去了人脸检测的步骤（图2），又因为我将上一步手部定位的矩形框信息作为输入，所以我还增加了一些步骤来对矩形框范围做出调整。除此之外，针对握拳这一手势状态，我也采取了椭圆拟合的方式降低手势误判的概率。最后再结合对轮廓信息的凸缺陷检测方法，就可以实现多种手势的判别了。

3. 手部定位

对手部进行定位其实可以归类为一个比较典型的目标检测问题，目前在目标检测领域比较成熟的方法是使用机器学习的方法。机器学习的方法好处在于，它可以避免由于复杂的背景等环境因素的变化而引入的一些干扰问题 [3]，也可以省去复杂方法的编写，只需要提供足够多的可靠的数据集，即可获得远超传统方法的识别率。当然，数据集的获取也就成为了这一类方法中比较困难的一个环节。

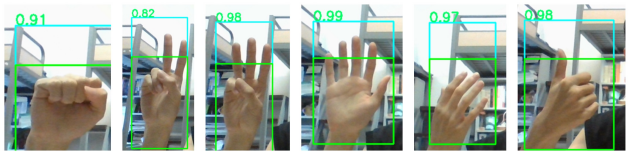


图 3. 神经网络方法对手部进行检测的效果，多种手势（包括未定义类别的手势）均被成功检测出来。

训练所用的数据集. 在 Dibia[3]的项目中，进行机器学习所使用的数据集来自于 EgoHands[1]，该数据集包含多种以谷歌眼镜拍摄的第一人称视角的图片，每张图片中均含有对于人手的标注，人手的姿态也多种多样，因此可以较好的应对电脑摄像头视角下的识别这一应用场景。

神经网络的结构. Dibia[3]在训练神经网络时采用的是预训练好的 ssd_mobilenet_v2_coco 模型。该预训练模型基于 SSD 结构，训练时只需用数据集对后几层的参数进行训练即可实现自定义的目标检测，我在实际使用时是直接使用完全训练好的模型，该模型的检测效果可以达到预期要求（如图3）。

4. 手势判别

人们区分不同手势的依据，其实主要来自于手势的几何轮廓信息，人类通过每根手指的张开状态 K，就可以完全判断出手势所属的类别 T ($T \in \{G0, G2, G3, G5, \dots\}$)。以 G0 和 G2 手势为例（手势的定义见图1），只要看到五根手指均为未张开状态，则可断定手势为 G0，只要看到手部的食指和中指处于张开状态，而其它手指处于未张开状态，则可断定手势为 G2。

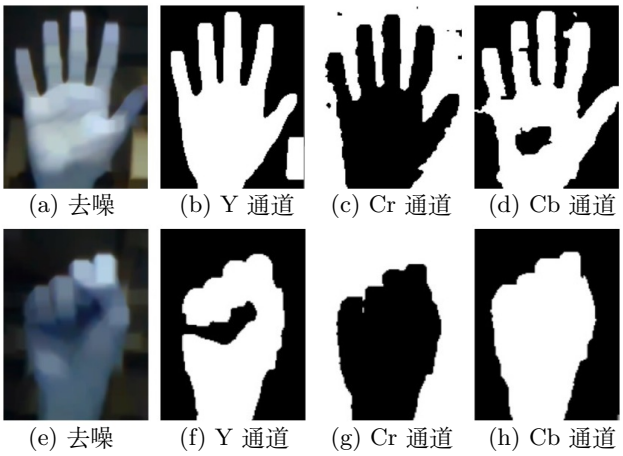
既然要从上一步手部定位所获取的手部图中进一步获取几何轮廓信息，那就意味着还要对图片进行加工处理，因为上一步输出的图片仅仅为裁剪后的区域图片，没有任何的附加信息。

手部图像的预处理. 预处理的第一步是对图像进行去噪，我首先是对输入的图片进行高斯滤波，接着采用了形态学开运算的方法进一步将图片模糊化，经过以上两个步骤的处理后可以忽略掉一些细小的噪声，得到的结果如图4中的 (a) 和 (e) 所示（由于使用的是 OpenCV 函数库，RGB 彩色图片未进行 BGR 的转换），可以看到图片的主体信息仍然得到了保留，而诸如手部的纹理和背景的图案等一些细节则被忽略掉了。

去噪之后需要对图像进行二值化，我采用 YCrCb 颜色空间分割的方法。这种方法已经被很多文章所采用，并且在早期的肤色分割问题上取得了比较理想的效果，根据 D. Chai[2]的研究，人体的肤色在 YCrCb 颜色空间的分布如下：

$$\begin{cases} 133 \le Cr \le 173 \\ 77 \le Cb \le 127 \end{cases} \tag{1}$$

在实际应用中，我并没有采用设置固定范围的方法对通道进行二值化，而是使用 OTSU 自适应阈值分割算法对 Y、Cr、Cb 通道的色彩值进行了分割，为了对比 Y、Cr、Cb 三个通道的分割效果，我对三个通道分别采取了 OTSU 算法，最终结果可以在图4中得到体现，我发现 Y 通道和 Cb 通道的阈值分割虽然在某些情况下也能取得比较好的二值化效果，但是随着环境和光照的变化也可能会分割不完整的问题（这可以从 (b) 与 (f)、(d) 与 (h) 的纵向对比图中看出），因此我得出结论，采用 Cr 通道进行阈值分割的二值化效果最好。



(a) 去噪 (b) Y 通道 (c) Cr 通道 (d) Cb 通道

(e) 去噪 (f) Y 通道 (g) Cr 通道 (h) Cb 通道

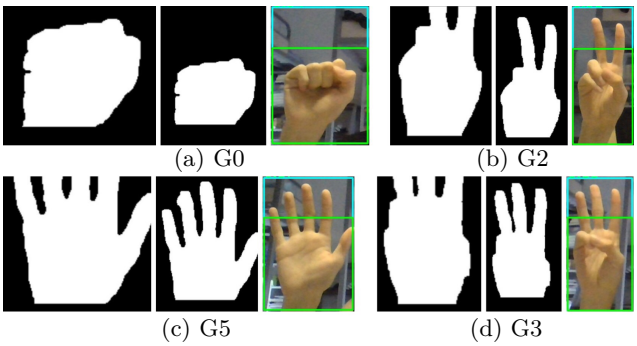
图 4. 手部图像的预处理。

二值化后紧接着的步骤是形态学开运算，这一步是为了消除二值化过程中产生的孤立噪声点（譬如图4(c)中存在的黑色噪声点）。

矩形范围的重调整. 由于使用的是别人训练好的神经网络，数据集的标注也仅仅以手部的整体位置为标签，因此手部定位这一步骤返回的矩形框只是框住了一个

大概的以手部为中心的矩形范围，这个范围可能并未包含所有有利于手势分析的图像区域，甚至有可能未将部分手指的延申区域考虑进去而导致有用信息的丢失。从图5中可以看出，手势 G0 的矩形区域比较合理，无需调整，但是手势 G2、G3 和 G5 的手指部分则落在了矩形区外部，因而有必要进行重调整。

为了解决这个问题，我提出了一种扩展图的方法作为手势判断的关键性步骤。所谓扩展图，就是将手部定位步骤输出的矩形范围进行扩展（这里我采用向上扩展的单方向扩展），扩展操作对于手势 G0 的判断来说可能是一种不利的操作，因为会增大噪声引入的几率（这个问题在下一小节中将得到解决），对于 G2、G3、G5 来说，扩展图无疑带来了巨大的帮助（见图5，扩展图使得原本被遗漏的手指信息得以恢复）。

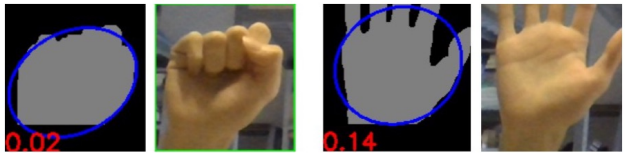


(a) G0 (b) G2 (c) G5 (d) G3

图 5. 四种手势未扩展和扩展后的矩形范围对比。彩色图中显示的绿色框为原手部定位输出的矩形框，亮蓝色的边界框为扩展后矩形框。扩展后的矩形框对于手势轮廓检测的改观很大。

同时，考虑到每次待识别的手势画面是未知的，即无法事先决定矩形框的大小是否应该扩展，那么在图像预处理时，就需要做两手准备，在实现的时候，我对扩展图和为扩展图均进行相同的操作，将两幅图（扩展图和未扩展图）的预处理结果共同传递给下一步。

椭圆匹配. 在上一小节中我们便看到手势 G0 具有的特殊性，该手势的定位矩形框包含了整个手部（手腕桡骨远端的部分）。特别的，该手势的外部轮廓没有大幅度的凹凸变化，大致呈椭圆状，这里通过与椭圆进行轮廓的匹配并计算差异值的方法便可以判断出手势是否属于 G0，当匹配后的差异值小于 0.03 时，判定手势属于 G0，反之则判定不属于 G0，这样可以有效地辨别 G0 和其他手势（如图6，G5 匹配所得的差异值远远高于 0.03）。



(a) G0 的差异值: 0.02 (b) G5 的差异值: 0.14

图 6. 不同手势轮廓与椭圆匹配的差异值。

基于手势所属的类别，可以决定手部图是否需要扩展：当手势初步判定为 G0 时，可以不进行图的扩展；当没有把握认为手势属于 G0 时，需要对图进行扩展以进一步分析轮廓信息。

凹陷点检测. 手势识别领域对于手势的分析有一套成熟的方法，该方法使用凸缺陷检测对手势的轮廓进行细致的分析，这可以在 Yeo[4]的论文中看到。而所谓凸缺陷，就是指凹陷，使用 OpenCV 中的 convexHull 函数可以检测出轮廓中的凹陷区域（也叫凸包），对凹陷区域再执行 convexityDefects 函数运算，就能从返回值中得到凹陷区域的起点、终点、最远点以及最远点的距离。最终我们所需要的信息就来自于最远点。

由于我所识别的手势种类不算多，所以没有采用复杂的分析策略，而是将手势识别的步骤简化为检测凹陷点的数目。我们知道手指张开时会与手掌部分形成一个凹陷区域，每当有一个手指张开时就会新增一个凹陷区域，每个凹陷区域又对应存在一个凹陷点，因此我们可以通过对于凹陷点进行计数，来简单判断手指的张开状态 K（见第4节）。

但是由于噪声的干扰以及手部线条的非均匀性，使用 OpenCV 函数进行检测时会检测出一些不希望判断为凹陷的位置，这样我们就需要添加一个判断条件，这里我通过“最远点距离”这一信息来对凹陷点进行筛选，当距离小于手部区域高的 8% 时，就认为该凹陷相比于手部轮廓的尺度过小，不满足判为凹陷点的条件。

当然正因为没有制定复杂的分析策略，所以这种凹陷点计数的方法仅仅适用于目前手势种类较少的情况，当手势种类增多时，需要更新这一策略来提高识别精度。

决策输出. 最终，通过椭圆匹配的差异值和凹陷点的数目这两个条件的结合，我能够做出手势的判断，具体

的决策如图7中所示。

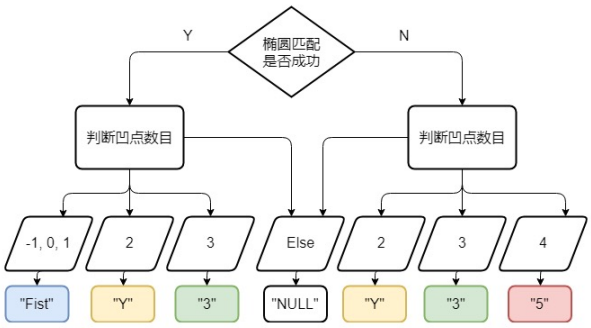
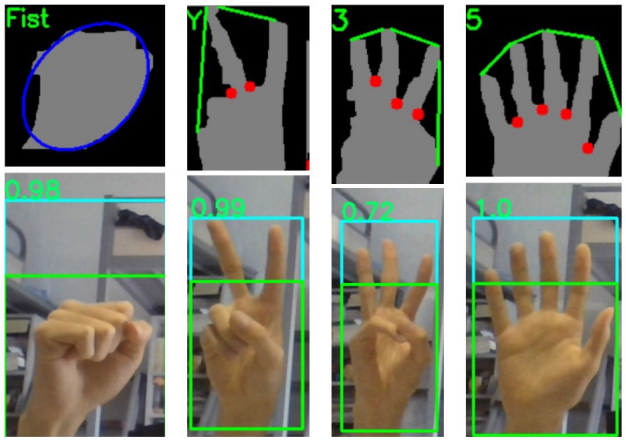


图 7. 决策树输出具体的手势类别。包括 G0、G2、G3、G5，它们的字符代表分别为“Fist”、“Y”、“3”、“5”，若判断结果不属于以上四种则输出“NULL”。



(a) G0 (b) G2 (c) G3 (d) G5

图 8. 四种手势的识别结果。

5. 实验分析

我对本文的方法进行了运行速率和识别准确率的测试，这两项测试的具体内容分别为：

- 运行速率：计算每秒运算的视频帧数目（FPS）
- 识别准确率：计算程序平均的识别正确率（AR）

其中 FPS 通过计算程序由开启到关闭过程中加载处理的帧数除以程序运行时间差即可得到。而为了定量的分析识别准确率，我定义了一个指标 AR，该指标的计算公式表示如下：

$$AR = \frac{60\text{ 帧中识别正确的帧数}}{60} \times 100\% \qquad (2)$$

另外由于本文的方法还未实现全部的功能，因此在测量 AR 值时的手势内容仅仅是从 G0、G2、G3、G5 四种手势中进行指定。最终实验测得的结果如下表所示：

手势类别	G0	G2	G3	G5
AR	95%	93%	60%	95%
FPS	12			

表 1. 实验测得的 FPS 和 AR。这里的测试环境是一台拥有 i5 1.6 GHz CPU，8GB 内存的机器。

从表1中可以看出我的方法对于 G0、G2、G5 三种手势的识别准确率较高，达到 90% 以上，但是对于 G3 手势的识别则不太乐观，仅仅只达到了 60%。通过分析，我发现了影响 G3 识别准确率的两个主要原因：一个是手部定位环节对 G3 的定位准确率较低（见图8(c)，G3 的定位准确率仅有 0.72），也就是说当手势呈 G3 状态时，所使用的神经网络模型无法成功判断这是一只手；第二个原因是进行凹陷点的检测时，由于 G3 手势的凹陷区域可能不太明显，导致凹陷点的漏检测。

另一方面，我的方法运行速率还算理想，在运行过程中对实验设备的 CPU 占用率不超过 75%，同时每秒处理的帧数大约为 12 帧，基本可以做到流畅的检测。

6. 总结与展望

我提出的基于图像处理和机器学习的手势识别方法，将目标检测的手段和传统手势分析的手段结合，通过缩小手势分析的范围，免去了人脸检测和背景移除等步骤，相比 Yeo[4]的方法，具有更强的适应性。

但是我的方法也存在着明显的不足，比如机器学习部分，训练神经网络模型所使用的数据集是基于第一人称视角拍摄的手部数据集，虽然能满足识别大部分的手势，但是对于某些角度或者某些光照条件下的手势，则会出现定位精度低的情况，因此未来我将考虑使用自己的针对性更强的数据集对网络进行训练，来避免定位失败的情况。另外，我也将细化手势判别的策略，通过增设更加详细的规则来提高手势判别的准确率。

最后，为了将本文的方法应用到人机交互上，我还需要考虑对手势触发器1以及手部的运动轨迹等进行分析。我也将继续探索机器视觉领域的其他方法来改善人类的生活，为生活带来便利。

致谢. 本项目的完成离不开「计算机视觉基础」这门课上老师的指导和帮助。

参考文献

[1] S. Bambach, S. Lee, D. J. Crandall, and C. Yu. Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In The IEEE International Conference on Computer Vision (ICCV), December 2015. 3

[2] D. Chai and K. N. Ngan. Face segmentation using skin-color map in videophone applications. IEEE Transactions on Circuits and Systems for Video Technology, 9(4):551–564, June 1999. 3

[3] D. Victor. Handtrack: A library for prototyping real-time hand trackinginterfaces using convolutional neural networks. GitHub repository, 2017. 2, 3

[4] H.-S. Yeo, B.-G. Lee, and H. Lim. Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware. Multimedia Tools and Applications, 74(8):2687–2715, Apr 2015. 2, 4, 5