数字图像处理 期末报告

王轲 14307130048

摘要

相比于完全把神经网络当成黑盒来做训练，本文尝试了一种混合的思路：先通过人脸特征点检测获得特征点，再通过特征点预估人脸位置、角度、眼睛开合度等参数，再通过一个 LSTM 网络进行参数的训练，并对视频做分类。

相关工作

2018 年 CES Asia 展会上，科大讯飞展示了他们的驾驶员疲劳检测系统。他们的系统能通过计算机视觉的方法，从摄像头中获取人脸朝向、位置、瞳孔朝向、眼睛开合度、眨眼频率、瞳孔收缩率等数据，并通过这些数据，实时地计算出驾驶员的注意力集中程度。我在现场体验了他们的系统，非常灵敏准确。

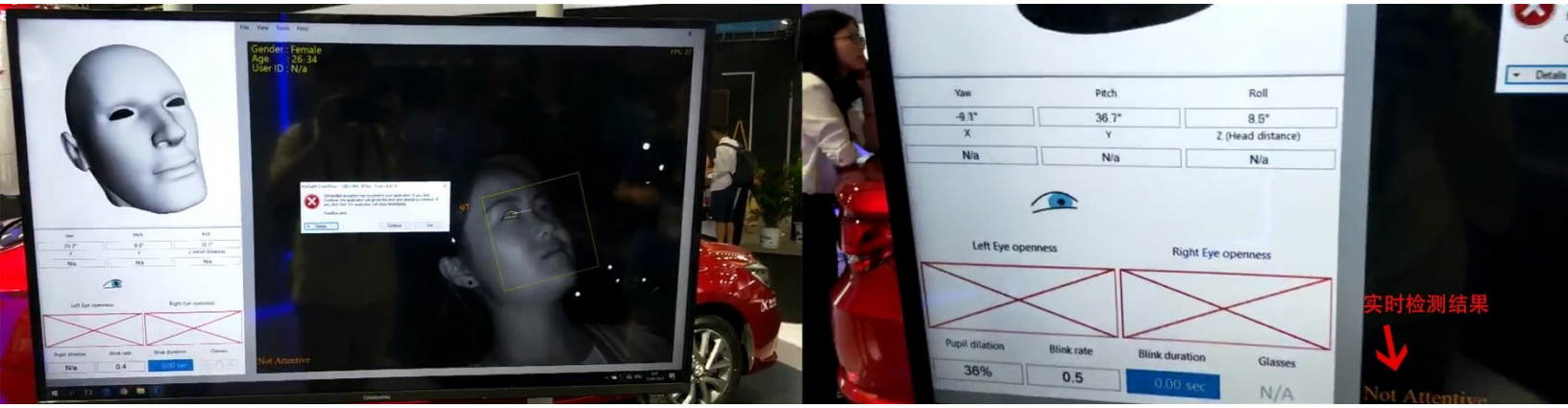

图 1 科大讯飞展会上展出的疲劳驾驶检测技术

总体思路

受到科大讯飞 DEMO 的影响，我做本期末报告时，也希望使用这种基于特征检测、参数计算的方法。我认为，这种方法不是完全黑盒的（相比于纯 CNN 网络来训练图片），在可解释性上可能会好一些，此外，这种方法还能够检测并记录别的体征数据，这些体征数据也有许多挖掘价值。

我做的实验分为以下三步：

1. 通过图片特征点检测的方法获得人脸特征点（如眼、鼻、嘴、轮廓）
   这一步可以使用著名的 dlib (King, 2018)来解决，有效果较好的已训练的模型：shape_predictor_68_face_landmarks.dat。使用它，可以得到 68 个人脸关键点。
2. 预处理，通过 68 个关键点判断人脸的朝向、位置、眼睛开合度信息
   判断人脸朝向和位置的主要难度在于估计人脸的三维信息，若有三维信息，就可以用 OpenCV 的函数 solvePnP，来计算出一个物体的朝向和位置。但是，因为单目摄像机的深度信息是缺失的，不能真正得到 3d 数据，所以在解人脸朝向时，需要配合一些人体测量学的统计数据（鼻根到人

脸各个器官之间的距离）才行。眼睛开合度的计算比较简单，因为已经有特征点数据，所以计算眼皮之间的高度，除以眼角之间的宽度即可。为了增加特征点数量，减少遗漏掉的信息量，68 个特征点也首尾相减（转换为位移），一并算入特征当中。

3. 对第二步得到的 6 个人脸纬度信息、2 个眼睛开合度信息，和 68 个特征点位移信息（每个特征点换为 x、y 位移两个 feature），一幅图 144 个 feature，放进 LSTM 神经网络中进行训练，并预测。

对图片进行人脸特征点位置检测

人脸特征点检测用到了 dlib，dlib 有两个关键函数：

dlib.get_frontal_face_detector()和 dlib.shape_predictor(predictor_path)。

前者是内置的人脸检测算法，使用 HOG pyramid，检测人脸区域的界限（bounds）。

后者是用来检测一个区域内的特征点，并输出这些特征点的坐标，它需要一个预先训练好的模型（通过文件路径的方法传入），才能正常工作。

使用预训练好的模型 shape_predictor_68_face_landmarks.dat，可以得到 68 个特征点位置的坐标，连起来后，可以有如图所示的效果（红色是 HOG pyramid 检测的结果，蓝色是 shape_predictor 的结果，仅把同一个器官的特征点连线）。

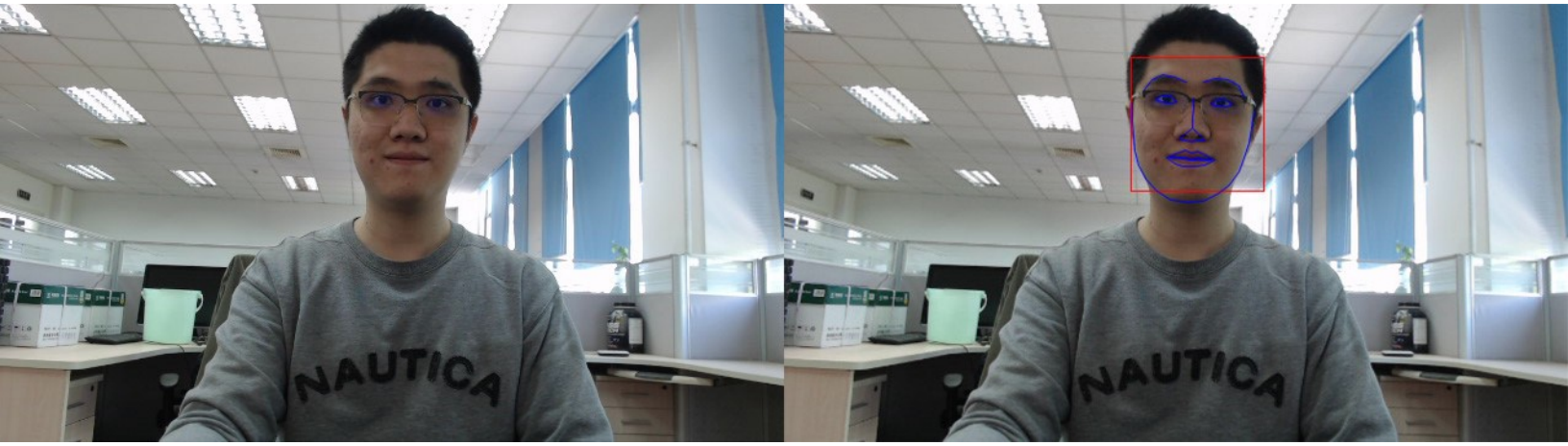

图 2 人脸特征点位置检测

对特征点进行预处理，得到人脸 6 维信息、眼睛开合度

疲劳驾驶打盹时，人脸会朝下垂，有时会轻微晃动，眼睛会微眯。这和人清醒时目光向前或略微向上，头部稳定转动很不一样。这是我们对疲劳驾驶直观的理解。这些信息放入 LSTM 网络里训练，让机器自动辨别这些信息，有可能能达到很好的效果。因此，从图片中采集这些信息很重要。

先讨论人脸 6 维信息的获得。如前文所述，单目摄像机不含有深度信息，单目的图像信息是无法估算人脸朝向的（因为相当于 3 维坐标投影到 2 维平面上，无法还原）。要把 2 维信息近似还原成 3 维信息，需要一些额外的信息（或先验知识），如人体测量学中的人脸五官平均距离 (Wikipedia, 2018)。

这里参考了一篇论文 (Lemaignan S. G., 2016)，和他的代码实现 (Lemaignan, 2018)。代码结合 OpenCV 和 Wikipedia 给出的人脸五官距离平均值，来对人脸进行建模。我基于作者的代码进行了改造，使其可以

批量预处理数据集中的视频截图，并以数组的方式输出人脸 6 维信息数据。具体的改造和代码运行方法请看我提交的源码。

使用本算法批量处理数据集中的所有图片。处理完成之后，便额外得到了 6 维特征，效果如下：

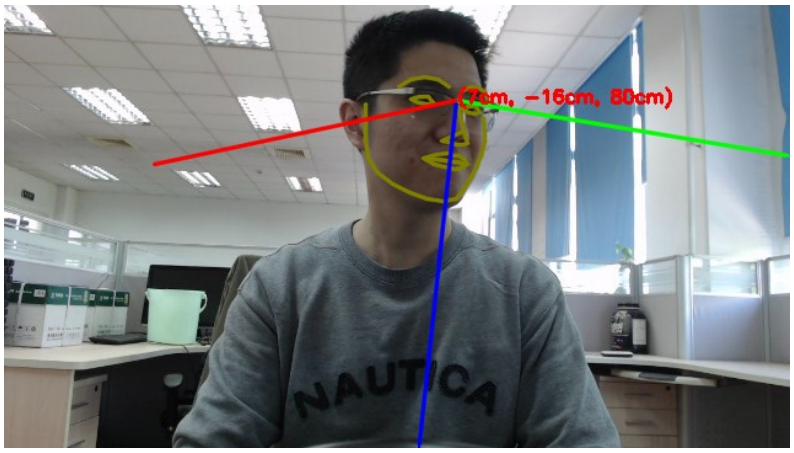

图 3 人脸 6 维信息的计算

接下来，计算眼睛开合度信息。68 特征点数据分布如图所示 (Rosebrock, 2017)：

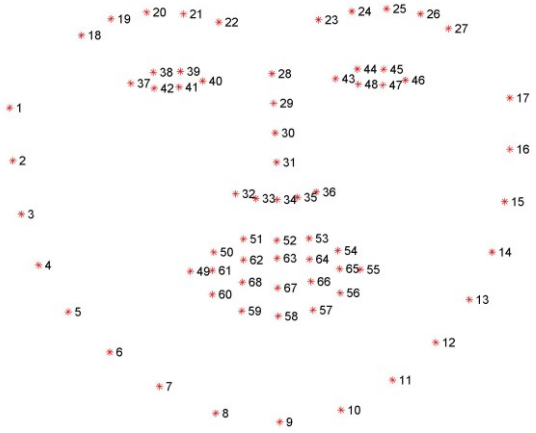

图 4 脸部特征点编号图

右眼开合度可以通过以下公式得到（左眼同理）：

$$EyeOpenness_{Right} = normalize\left(\frac{P_{42}.y + P_{41}.y - P_{38}.y - P_{39}.y}{P_{40}.x - P_{37}.x}\right)$$

头部 6 维信息、加眼睛开合度两维信息，共 8 维信息。如果有可能，还应该采集瞳孔朝向、瞳孔收缩率等信息，但我并没有能成功。如前文所述，一副图片只抽 8 个特征，信息丢失可能比较多，因此，我把 68 个坐标点进行一下处理，也作为特征加进去。考虑到坐标本身意义并不大，坐标位移意义反而大一些，我把这些坐标点首尾相减，换算成了位移。一个位移有 x、y 两个分量，所以这又是 136 个特征。这样，预处理就完成了。把一张图抽取为 144 个和脸有关的 feature，可以屏蔽掉许多图片细节，如光照、肤色、头的位置（和身高有关）、背景里窗帘的飘动……（尽管可能丢掉有用的信息）。更加重要的是，

一张图片变为 144 个数字，大大简化了输入数据，每一帧输入数据变简单，就让处理帧和帧之间关系（时序信息）变得可能（计算量不至于过大）。

训练 LSTM 网络模型并进行分类

我大学期间并没怎么接触过机器学习，所以这一步更多是依样画瓢，理解不深入。我进行了一些调查后，发现了 LSTM 网络，可以用来处理时序信息。我尝试了 Tensorflow，但感觉还是比较难上手，后来切换到了 Keras，用了 Keras 封装的更加高层、易懂的 API。

Keras 的 LSTM 层输入要求是三维的数据：（样本编号，时间帧编号(timestep)，特征向量），预处理完的数据正好是这样三维的，大多数的 shape 为：(?, 64, 144)。

我基本就是参考 Keras 文档中的示例代码，搭建了一个 LSTM 网络 (Keras, 2018)。在 LSTM 层后，添加了一层 Dense 层，模型就搭建完成了。

训练时，我跑了 500 个 epoch，最终结果为：测试集上 78.12%的正确率。而且因为把图片都抽成特征了，所以训练的速度非常快。效果还是比较可喜的。

不足之处和提升空间的讨论

1. **LSTM 的使用可能不足**
   我对机器学习方面的知识所知甚少。特征提取完之后，我在模型搭建和训练这一步做得很有可能不好。如果有机器学习方面才华横溢的人能指点一下，结果很可能能有所提升。也可以试着把 LSTM 换成别的，如 SVM、简单 RNN 等（疲劳驾驶检测其实可能不需要 Long-term memory）。
2. **单目摄像机获得信息少，人脸信息估计较差**
   因为人脸信息估计用到了统计学数据，但这个数据不是很精确，所以最终得到的角度估计效果并不是很好（一些帧存在跳跃现象）。换成双目摄像机，或者 RGB-D 摄像机，相信特征点提取的效果会好很多。如果疲劳驾驶检测要被广泛投入生产应用，那么升级硬件设备是可行的。
3. **基于人脸检测的图像处理健壮性略差**
   用 HOG pyramid 方法，一些帧中检测不出人脸（可能因为光照、帽子等原因），虽然这些帧不多，但是它们对整体效果可能有比较大的影响。这些帧，我目前的处理方法是跳过（可能导致时序上不连续）。也可以尝试使用插值的方法来弥补缺失的帧。
4. **特征抽取仍然不足**
   科大讯飞做的系统，可以抽取图片中眼球方向、瞳孔收缩率作为特征。然而本文并没能够抽取出这些特征，因此可能遗漏掉关键的信息。

总结

本文尝试了一种基于特征和参数估计的方法，最后一步分类时，仍然用到了深度学习的方法。受限于个人技术能力，许多可能可以进一步提升准确率的工作还没有做，本文分析了这些可能的改进策略。总体而言，实验结果还是不错的，准确率较高，模型训练快速。这足以说明，先提取特征点再使用机器学习的方法，是一条值得探索的路。

REFERENCES

Keras. (2018, Jun 8). *Sequence classification with LSTM*. Retrieved from Keras Documentation: https://keras.io/getting-started/sequential-model-guide/

King, D. E. (2018, May 26). *dlib*. Retrieved from dlib C++ Library: http://dlib.net/

Lemaignan. (2018, Jan 8). *3D head pose estimation using monocular vision*. Retrieved from GitHub: https://github.com/severin-lemaignan/gazr

Lemaignan, S. G. (2016). From Real-time Attention Assessment to “With-me-ness” in Human-Robot Interaction. *Proceedings of the 2016 ACM/IEEE Human-Robot Interaction Conference.*

Rosebrock, A. (2017, Apr 3). *Facial landmarks with dlib, OpenCV, and Python*. Retrieved from pyimagesearch: https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

Wikipedia. (2018, May 9). *Human head*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Human_head