**数字图像处理大作业报告**

**姓名：高子琪**

**学号：2018210377**

**专业：精密仪器系**

# 1 题目研究背景及本文摘要

在汽车行驶过程中，驾驶员疲劳驾驶成为众多交通事故的隐患因素之一。据统计表明，大约15％～20％的致命交通事故由疲劳驾驶导致。驾驶环境的随机性、复杂性等特点使得一般的检测手段难以对驾驶员的疲劳状态做出有效的评估及预警。 本作业为特定环境下进行疲劳驾驶检测中的面部定位和疲劳状态的判定。进行的项目分别为睁闭眼检测、张闭嘴检测、扭头与否检测和打电话检测。使用了matlab和python等语言编写程序，在老师所给的图片集中能做到较好地泛化，有一定的准确率。

# 2 作业任务简述

1. 作业包提供了视频文件，截取视频为图片集，直接对图片进行检测。
2. 使用方法和原理要有理有据，使用所学图像处理知识并进行延伸，把自己的理解体现在作业中。
3. 进行睁闭眼检测，在拥有图片集中尽量泛化，检测睁和闭两种情况。
4. 进行张闭嘴检测，检测张嘴和闭嘴两种情况。
5. 进行扭头检测，检测是否左右扭头。
6. 进行打电话检测，检测是否在打电话或将要打电话。

# 3 作业完成方法和结果

## 3.1睁闭眼检测

## 3.1.1预处理

首先使用软件KMPlayer将图片按照每秒五帧截取为图片。在所有图片中按照驾驶员和是否带眼镜分别取出一部分图片进行待处理，尽量做到各式各样的人和场景都有，以便随后选择方法和调节参数。

## 3.1.2二值化

首先对图像进行二值处理，尝试了大津法和迭代取阈值的分割方法，直接进行大津法的效果欠佳，因为是函数自动选取了阈值所以无法调节，不能符合自己的需求。迭代法选取的阈值效果较好，但是时间太长，这对实时检测是不利的，因此不符合要求。

因为这一步相当重要，可能所有项目都要使用，而且对以后的效果有很大影响。因此谨慎选择先进行聚类后进行大津二值化，首先聚类的类别数目可以自己选择，然后此种方法应用也比较广泛，经过尝试后比较符合本作业要求。

聚类程序为kmeans.m，二值化直接在主函数中进行，主函数名称为eye\_identification.m。

取几张图片进行二值化效果展示，效果如下图所示，选取聚类类数为3.

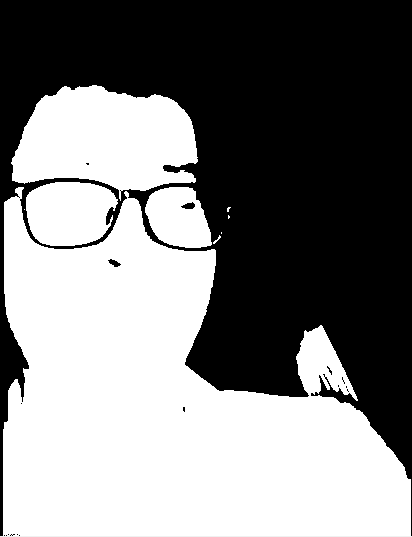


图1 二值化效果图

## 3.1.3人脸定位

这一步骤的目的是把人脸粗略地找出，是为了减小环境对之后的框出完整人脸的步骤的影响，不需要过于准确框出，但是人脸内容只能多不能少。

采取方法主要是求取每行像素值不间隔为1的最大值，例如该行像素值连续为1的部分为10，20，30，则选取30为该行的value。这样做的好处是尽量排除非人身体的像素，将每行的value组成一个列向量white\_row，对此向量进行操作来选取框出人面部的上线和下线。

对于上线的选取，从上向下扫描white\_row向量，找到第一个不为0的值，在此画线。对于下线，以类似的方法寻找，以white\_row的最大值进行参考设定阈值。

在上下线确定之后才可以确定左右线，这样很大一部分干扰就已经被移除了，此部分的参数设置较为复杂，阈值的设置也绞尽脑汁。效果图如图2所示。

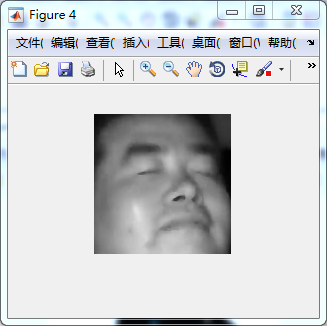
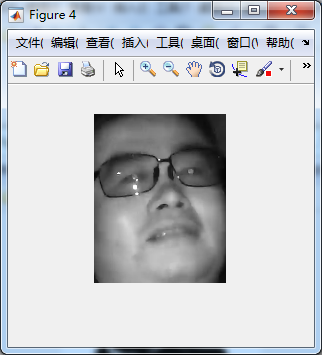


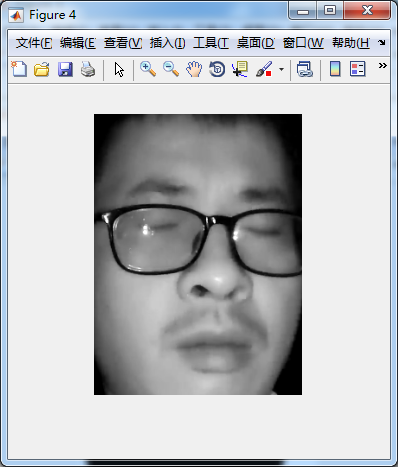
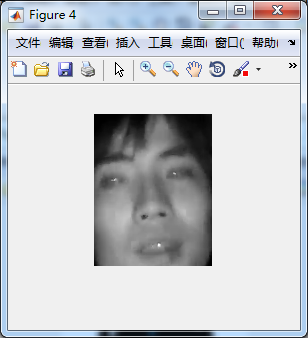


图2 脸部初步定位

接下来定位更加准确的面部，目的是将面部尽量恰好与矩形框相切，将面部充满整个框图，这样便于接下来的提取眼睛的步骤。

使用方法与上面类似，因为在图2的基础上提取，因此免除了很大一部分干扰，能较好地提取。提取效果如图3所示。





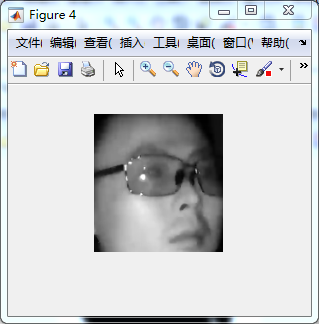
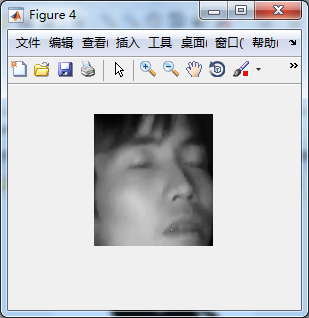


图3 更小面部提取

## 3.1.4人眼定位

尝试了多种方法进行人眼定位，其中首先想到的是质心法，利用人脸的质心在人脸中的位置规律寻求人眼的最大概率出现的位置，但是经过质心的定位后发现找不到有效的规律，而且眼镜会对质心有很大的影响，因此本方法舍去，程序保留，已注释掉。而质心提取的效果如图4所示。

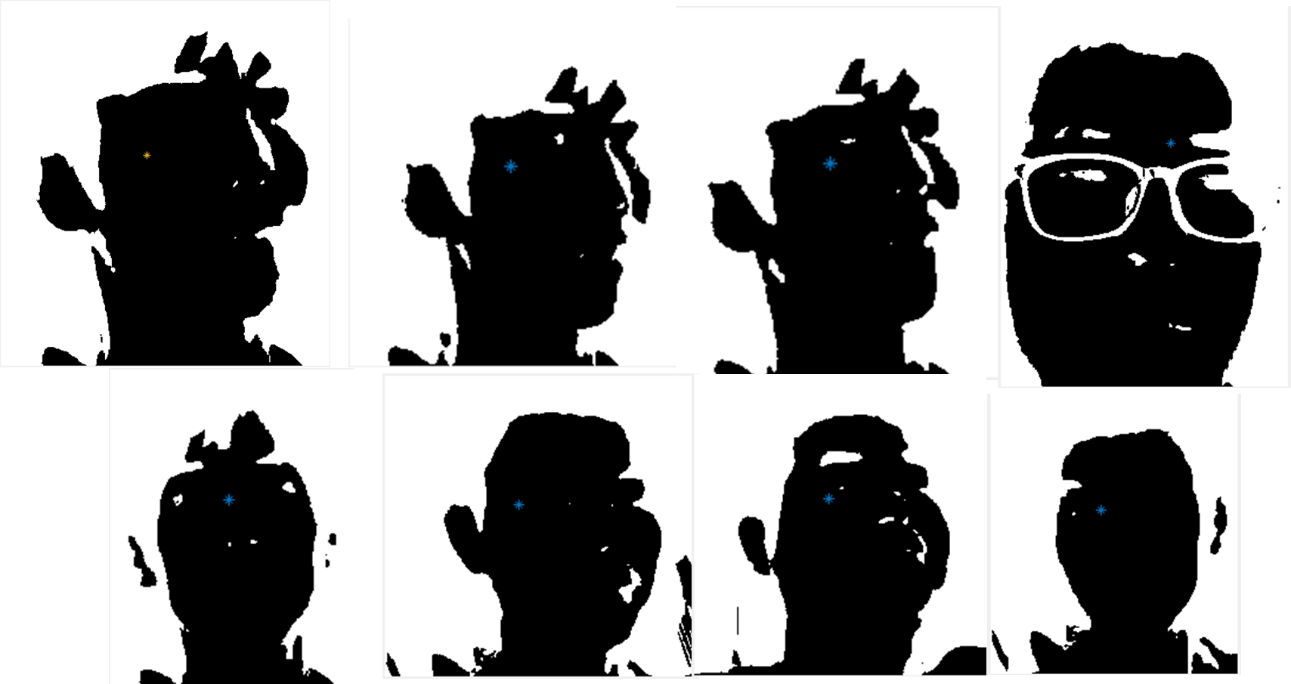
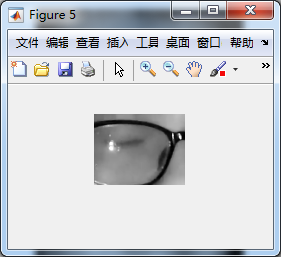
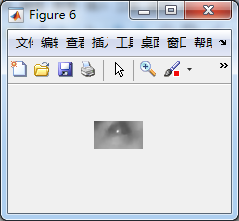
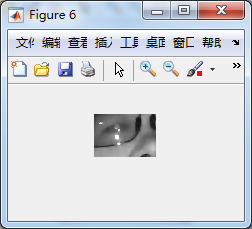
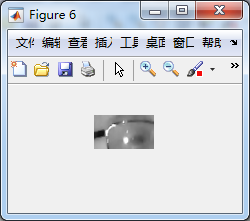
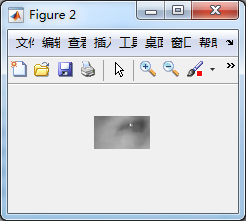
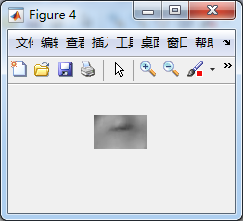


图4 人脸质心位置图

在这条路行不通以后就想到既然我已经把脸框出了，那就应该可以直接按照人脸分布比例直接定位眼的位置，在进行方法尝试和参数调节后发现确实可行。

因为在本作业中大部分人脸为侧视图，也有部分为正视，因此选定人的右眼为定位对象，在框选时也要注意两种情况下都能准确框选。人眼定位的效果如图5所示。





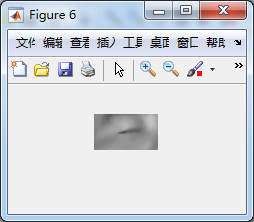
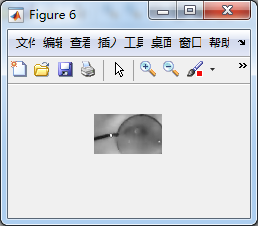
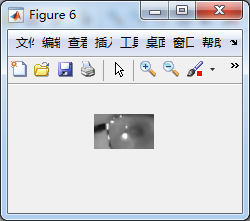
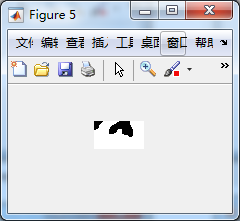
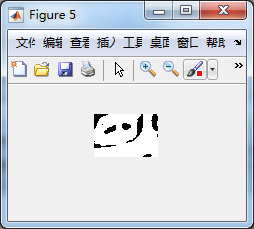
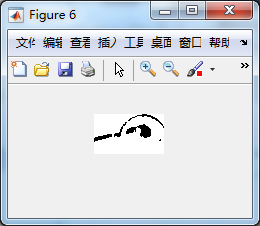
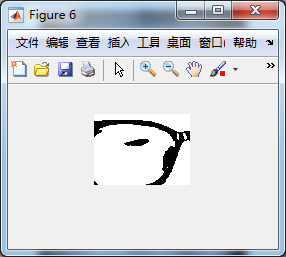
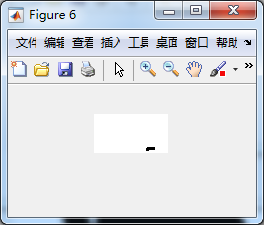
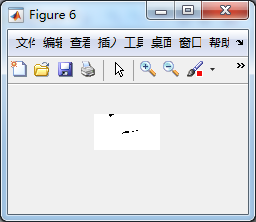


图5 人眼定位

## 3.1.5 人眼状态检测

首先采用如上先聚类后分割的方法进行二值化，选取类别数为2进行聚类，而后大津二值化，效果如图6所示。





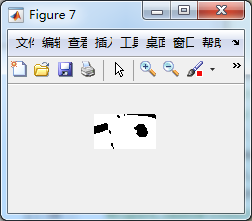
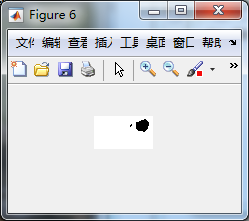
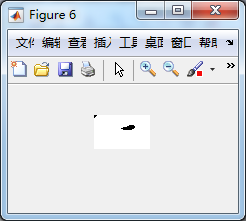


图6 人眼二值化

随后若想采用连通域像素值数目直接进行分类难度较大，因为有眼镜框的干扰，因此使用连通域方法进行眼镜的去除。首先发现眼镜框大多是比较长的连通域，首先对上图进行膨胀处理和一些开闭运算，使不连接的眼镜框变为连通域，然后除去面积过小的连通域和长度长于某阈值的连通域便可以得到眼睛的连通域，以眼睛的面积和长度作为特征设定阈值进行分类。

## 3.1.6 GUI封装后结果展示

对以上所有过程进行GUI封装后效果如图7所示，GUI程序为cin.m。

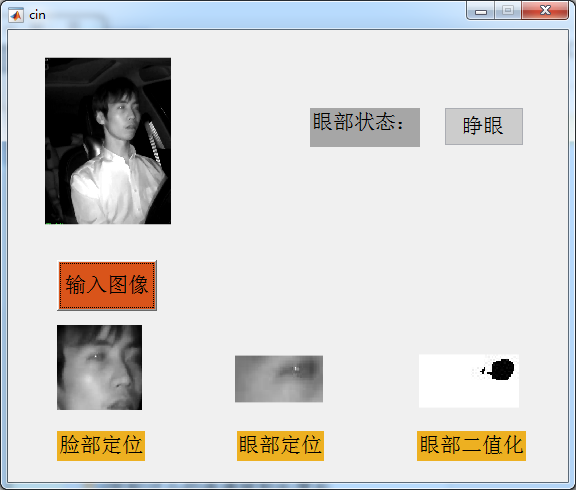














图7 GUI完整效果展示

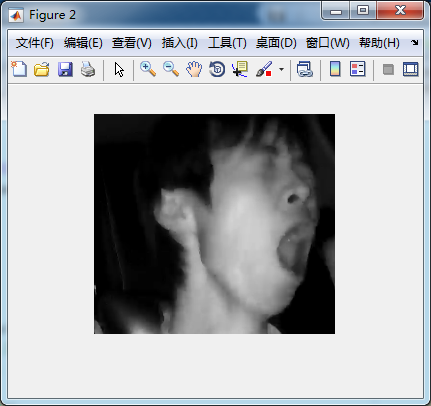
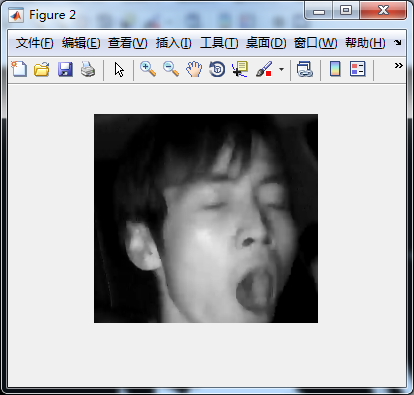
## 3.2 张闭嘴检测

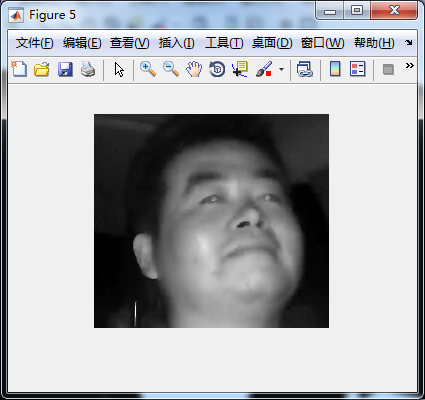
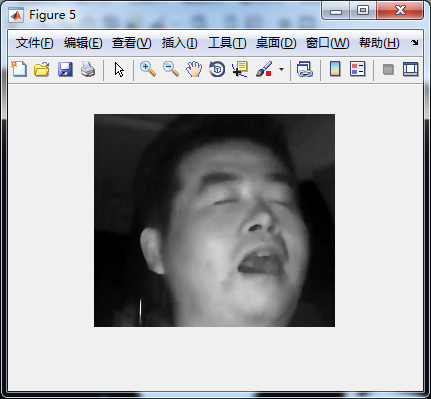
## 3.2.1 预处理

首先使用软件KMPlayer将图片按照每秒五帧截取为图片。在所有图片中按照驾驶员的位置和其他状态分别取出一部分图片进行待处理，尽量做到各式各样的人和场景都有，以便随后选择方法和调节参数。

## 3.2.2 二值化及脸部获取

二值化过程和3.1.2中二值化过程相同，聚类类别K设为3.脸部粗定位过程和3.1.3相似，调节了其中一部分参数使效果更佳。脸部截取效果如图8所示。





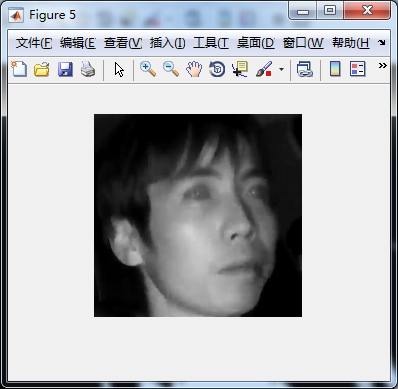
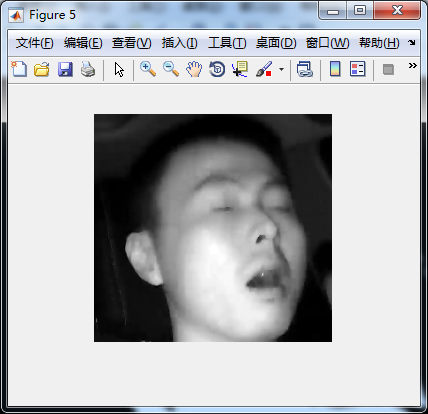


图8 脸部获取效果图

仍然首先尝试使用质心寻找嘴的位置，质心求取如图9所示，但是发现无规律可循，主要是因为脸的侧摆与否，正脸的质心和侧脸是不同的，因此无法使用质心找到嘴的位置。

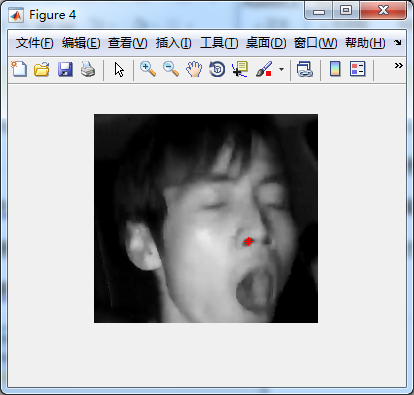
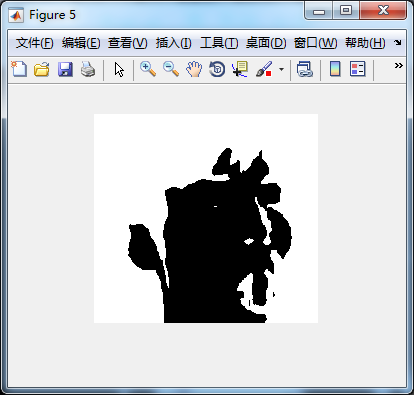


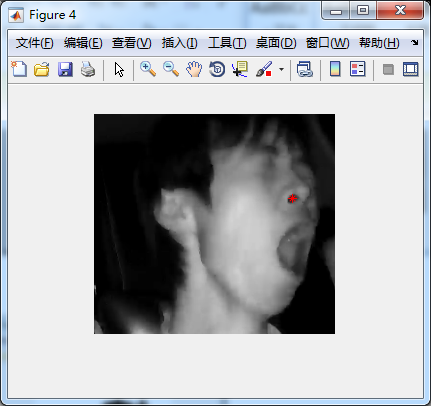
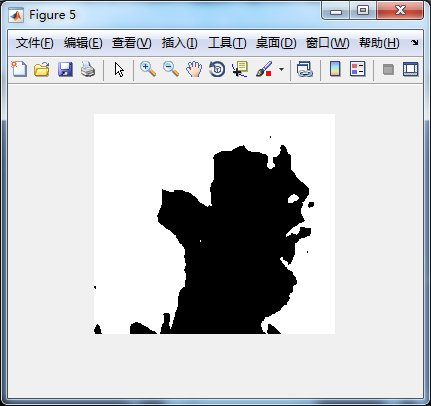
图9 脸部质心分布

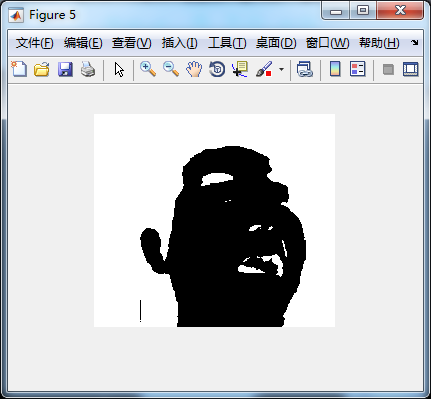
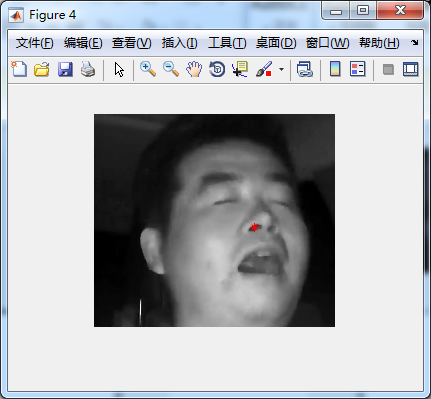
## 3.2.3 鼻孔定位

在舍去了质心参考定位嘴的方法之后想到观察如上图的二值图像，发现较多的人会有显著的鼻孔。首先考虑的是使用某个显著特征和此特征和嘴巴的位置关系区定位嘴，首先，眼睛较难提取到，因为周围有眼镜干扰，然后嘴巴的二值其实并不好，其连通域特征不够明显，然后鼻梁是时有时无，只有鼻孔的特征较为普遍和明显，因此尝试使用鼻孔去定位嘴巴。

首先对二值图像重新生成，生成质量较好的二值图，然后寻找其连通域，并求出每个部分的面积，首先去除面积过小的连通域，过大连通域先不急去除，此过程中使用regionprops度量区域属性，进行一些连通域的增加和去除，在大概的区域范围内定位鼻孔的位置，二值化效果和定位效果如图10所示。





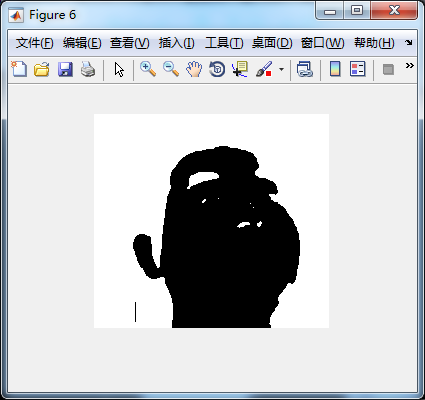
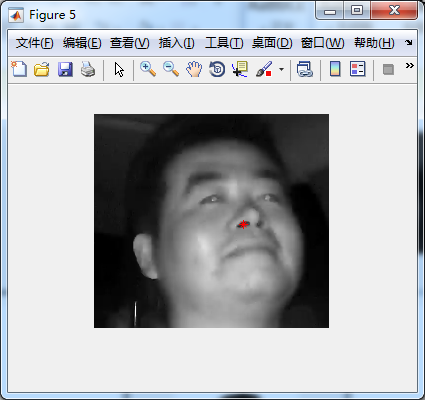
 

图10 二值化和鼻孔定位效果

## 3.2.4 嘴巴定位和状态判断

鼻孔定位之后，依靠先验比例和位置关系定位嘴，根据二值化的结果，如果未张嘴，绝大多数图片二值化后嘴部全为黑色，因此只需要取鼻孔下方一定区域内进行判断即可。检测成功与否的主要阶段是上面，此步较为简单。展示完整的GUI封装如图11所示。

嘴部检测的程序为mouth\_identification.m，GUI封装后的程序为mouth\_GUI.m。













图11 GUI封装效果图

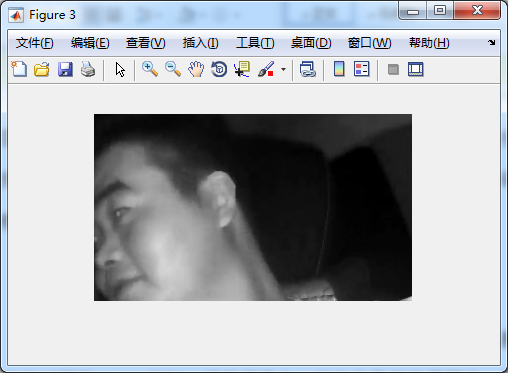
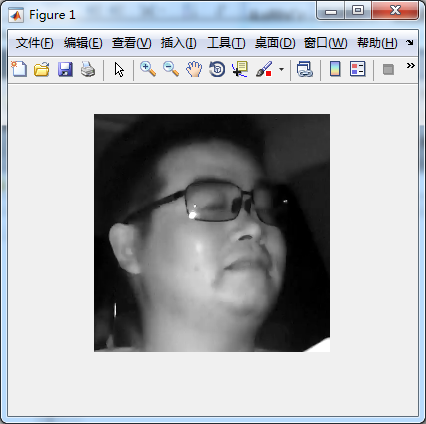
## 3.3 扭头检测

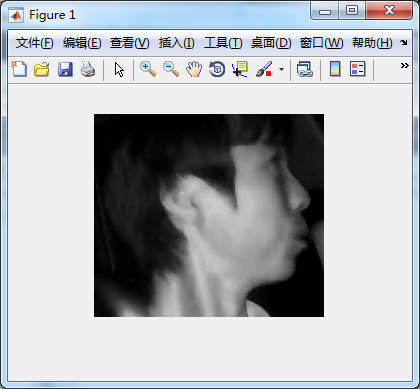
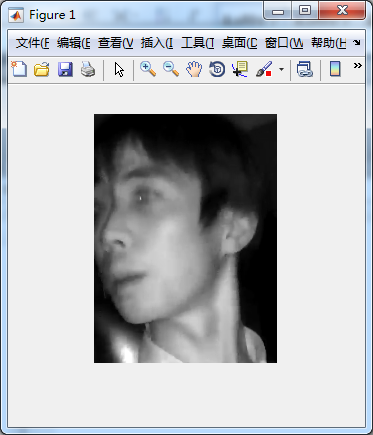
## 3.3.1 预处理

首先使用软件KMPlayer将图片按照每秒五帧截取为图片。在所有图片中按照驾驶员的位置和其他状态分别取出一部分图片进行待处理，尽量做到各式各样的人和场景都有，以便随后选择方法和调节参数。

## 3.3.2 头部定位

二值化过程和3.1.2中二值化过程相同，聚类类别K设为3.脸部粗定位过程和3.1.3相似，调节了其中一部分参数使效果更佳。脸部截取效果如图11所示。



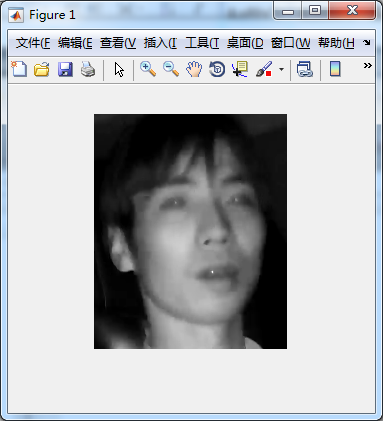
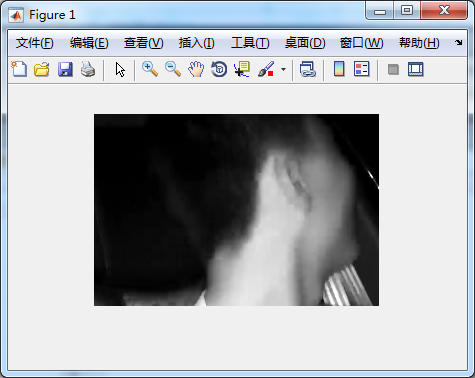
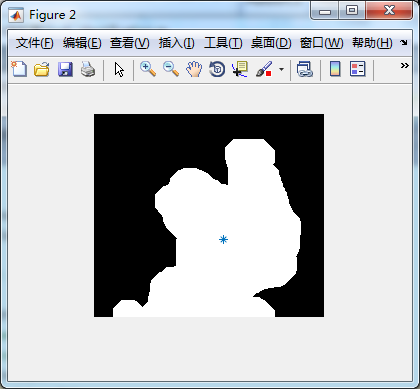
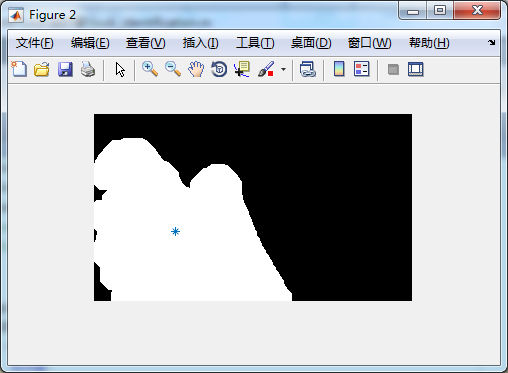
 

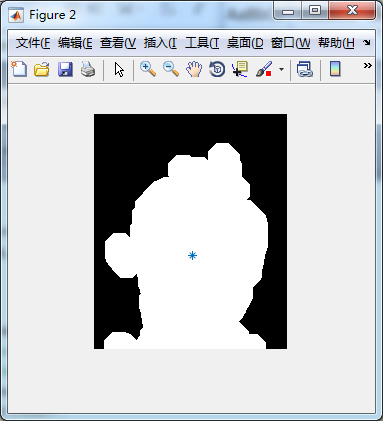
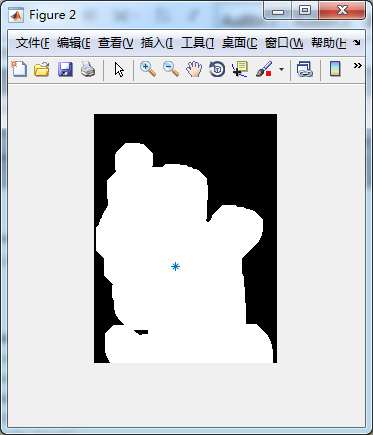
图11 头部定位和截取效果图

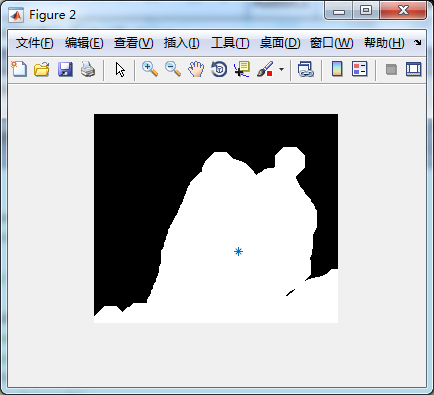
在此步骤中注意只需要进行粗定位和截取即可，因为每个人当发生扭头的时候身体和头的位置和姿态各异，而且不是正脸对着相机，所以比较难精确截取，而做到此已经足够为后面的方法服务了。

## 3.3.3 求取质心

首先进行二值化并将背景变为黑色，前景变为白色。然后使用直径为8的圆盘进行多次闭操作，弥合间断和沟壑。使用for循环在两个方向上求取质心坐标并标记出来，效果如图12所示。





由以上效果图可以看到，质心的位置大致在耳朵附近，其位置不像进行闭眼检测的时候一样飘忽不定，因此可以利用其做一些事情，首先我尝试了类似霍夫变换的操作，以质心为中点，画出不同角度对应的最远1像素到质心的距离图，生成直方图寻找规律。此方法的灵感来源于我发现大多数扭头的图片会出现很明显的V字形图样，而不扭头时一般白色图案较为规则，按照这种思路只要寻找直方图中的极小值在进行判断即可（猜想）。

扭头时直方图效果如图12所示，不扭头时直方图如图13所示。

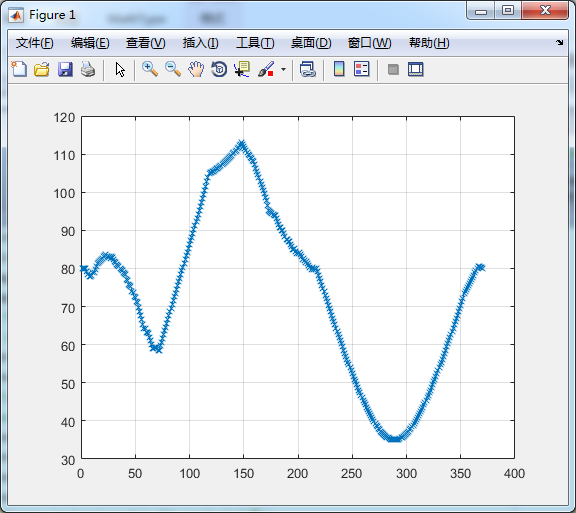
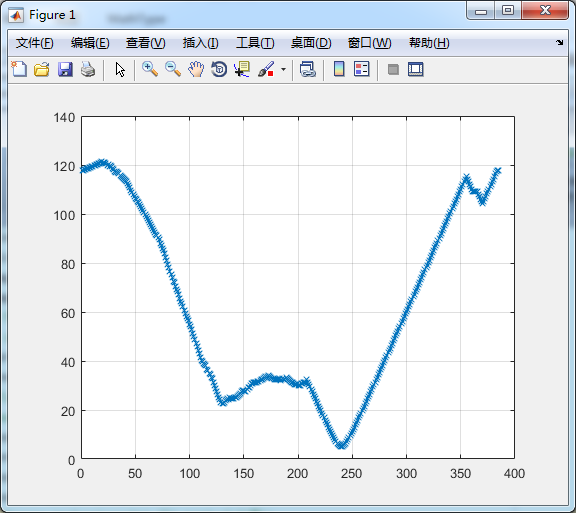


图12 扭头时直方图

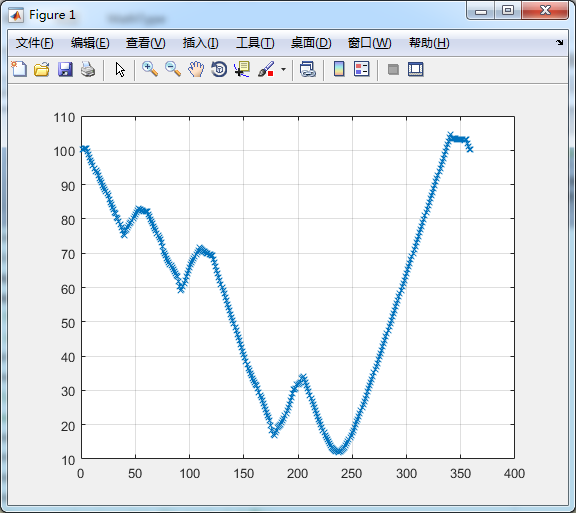
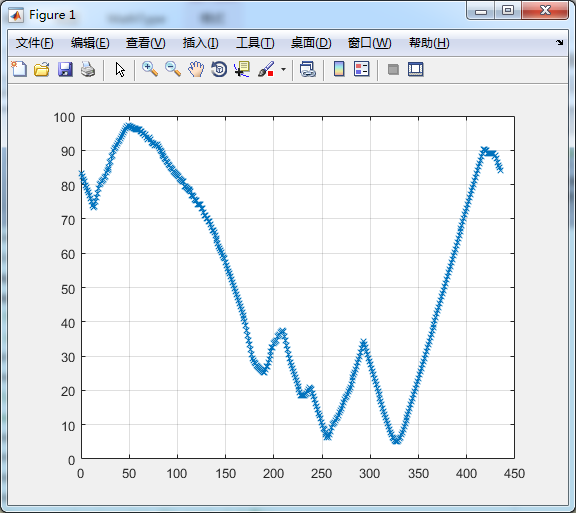
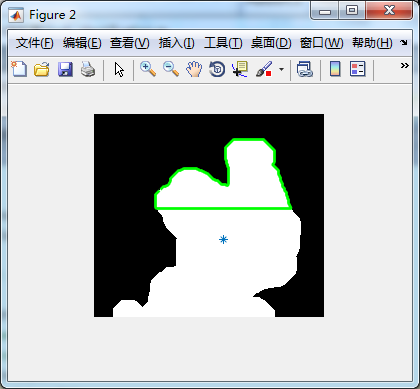
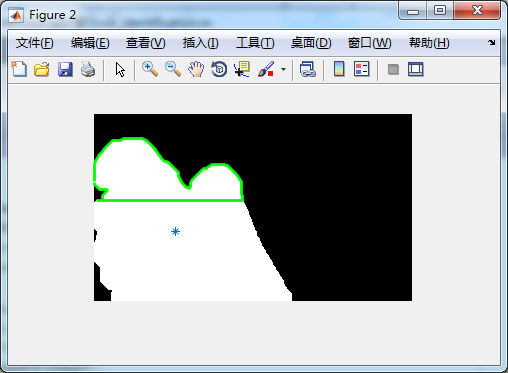
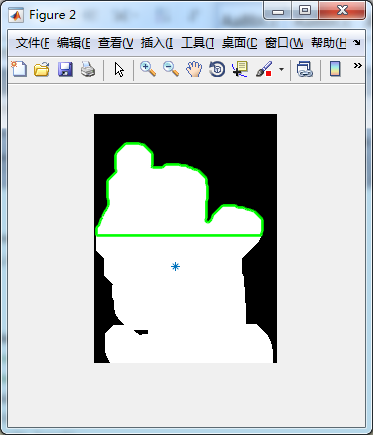
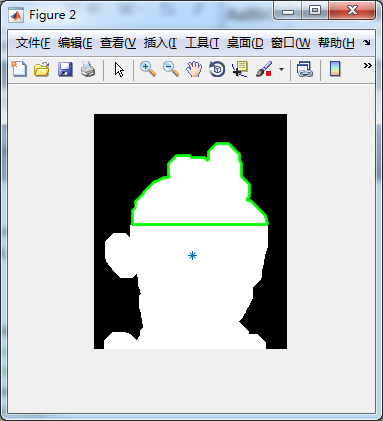
 

图13 不扭头时直方图

在进行了大量的统计实验后，发现扭头时，极小值点可能为1、2甚至为更多，但1和2比较多，而且若存在多个极小值时，其横坐标相差较大，利用这个特性去进行分类后发现效果不如人意，可能缺少了一些预处理或方法本身有问题。

在进行了一些思考过后发现既然是想通过头型的规整度来进行判断，那便可以使用一些度量形状的参数，寻找能否找出一些规律，于是边进行如下操作：首先是否扭头二值图中的头部形状的变化主要来自鼻子以上，下巴一下的部分会对任何度量方法造成干扰，因此首先选取以质心为参考的特定范围的区域进行截取，效果为图14绿色直线包围的部分。



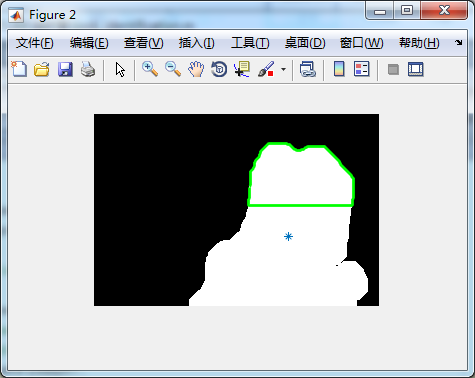
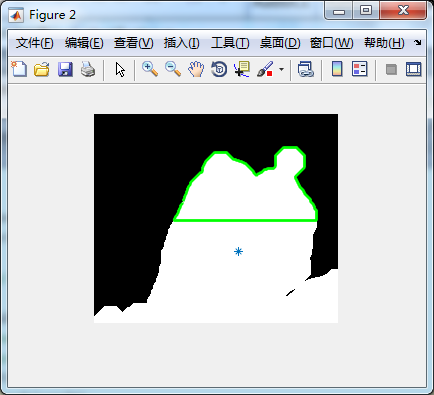
 

图14 选取特定区域

## 3.3.4 进行形状判断

进行形状判断时使用形状参数F进行描述，有F=，当连续区域为圆形时，F为1，当为其他形状F大于1，由大量的分析之后可以发现，扭头时，连通域形状很不规则，反之，形状的对称性和圆整性较好，因此F值较小，并以此来进行分类，经过大量实验，决定使用1.5为阈值进行分类，整体分类效果GUI封装如图15所示。大体效果良好但是存在一些误判。













图15 扭头检测GUI封装

## 3.4 打电话检测

首先，经历了几天的matlab基础方法的尝试后，发现有如下问题：打电话大体包括左手和右手两种情况，在右手情况下是比较容易通过连通域和形态学处理检测出来的。在左手情况下，安全带部分的连通域会严重干扰左手抬高的识别，并且左手是否有拿手机不是很清晰显示。因此在经历了几天的纠结之后决定选用CNN进行训练。

个人认为calling部分可以使用CNN是因为打电话这一行为引起的像素值和数量的变化在整幅图中占比较大，有一定的可用性。因此进行尝试。

## 3.4.1预处理

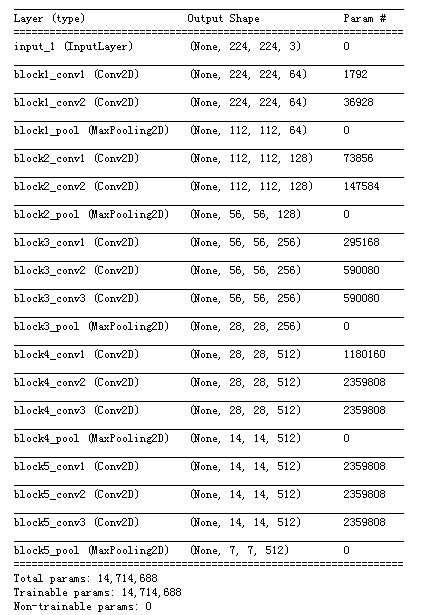
首先将视频以7帧每秒的帧率截图，存图，选取数据集的过程较为复杂，工作量也比较大，主要是calling和nocalling，train和val和test的比例大体为6：2：2，在train的calling里只选用截取图片中的举起手打电话和手拿手机放方向盘上将要打的情况，以此数量为基准设定其他集合的数量，在选定其他集合的时候注意尽量使用最多的人的类别和是否戴眼镜、是否歪头、是否张嘴，把尽量完全的情况包含进去。

## 3.4.2 使用基于keras的VGG16进行训练

程序保存为了jupyter文件VGG16.ipynb。

VGG16模型是图像分类问题中的经典模型，虽然以此进行二分类有一小些大材小用，但是只要结果够好，有说服力，便可以使用。

本文在VGG16基础上设计了新的全链接层并对VGG最后三个卷积层参数重新训练。整个网络构图如图16所示。



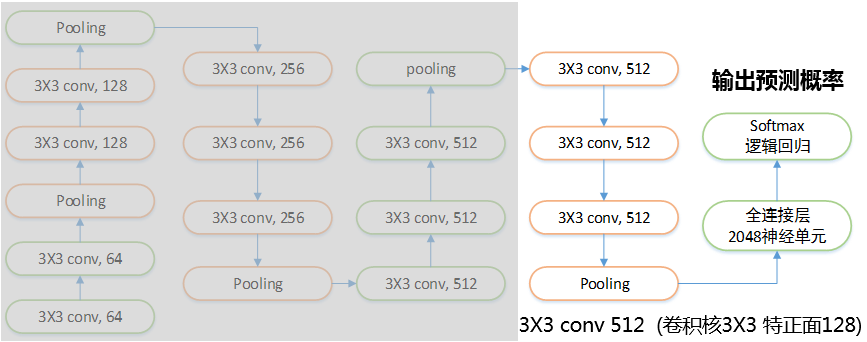
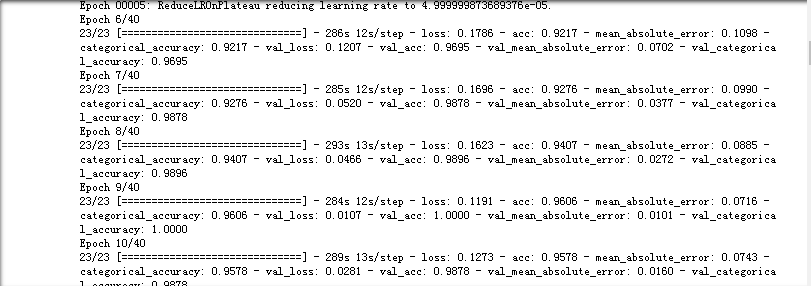


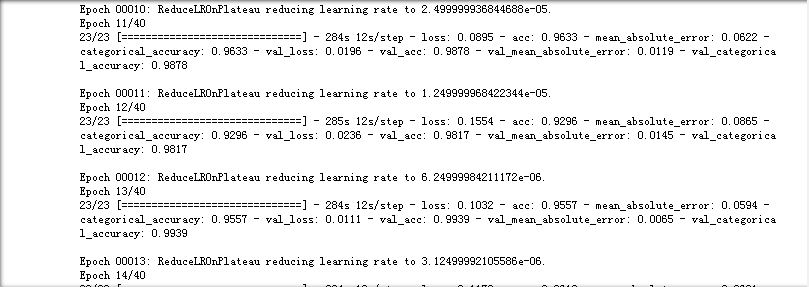
图16 改进后的VGG16结构图

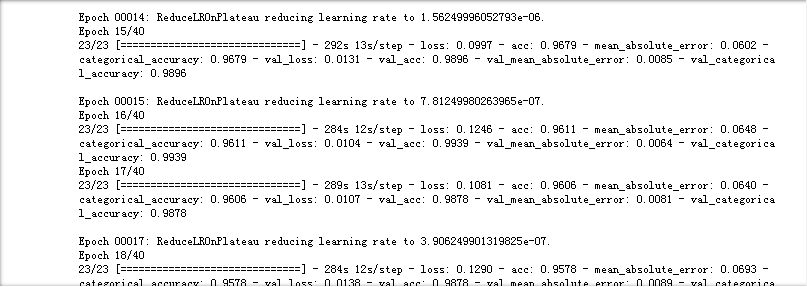
采用随机梯度下降法训练参数，训练40轮次，学习率初始值为0.0001，每轮训练大概280秒。在测试集上准确率约99%

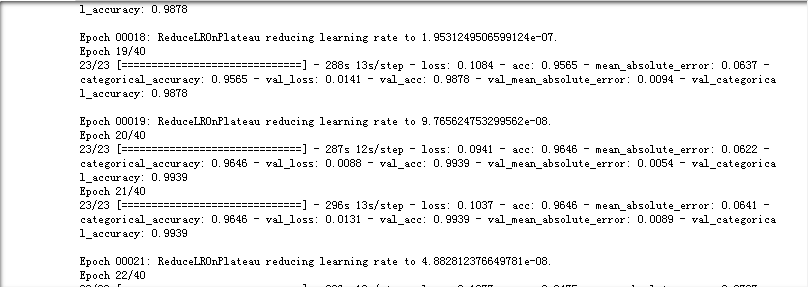
训练结果如图17所示。

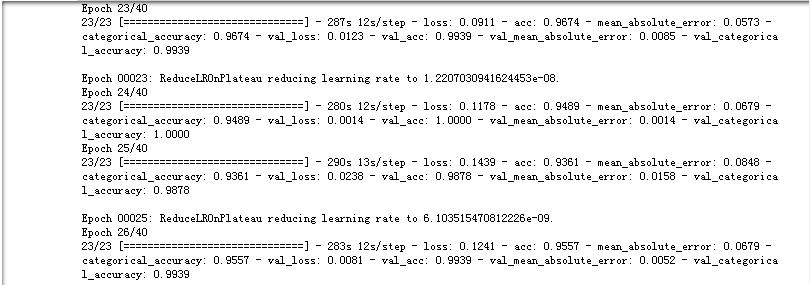


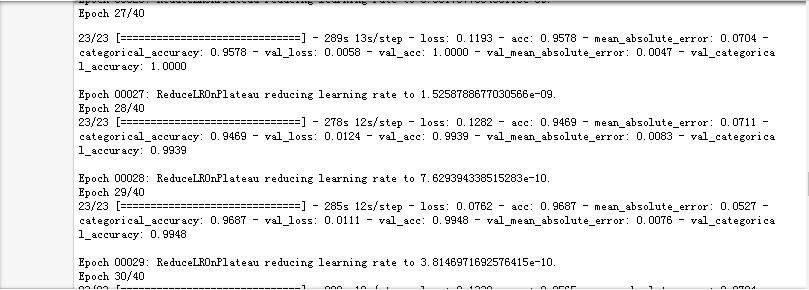


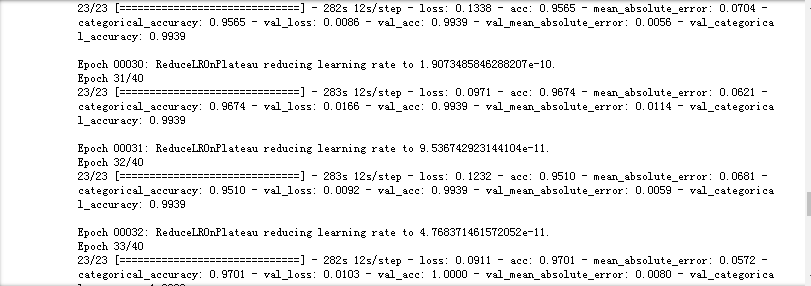


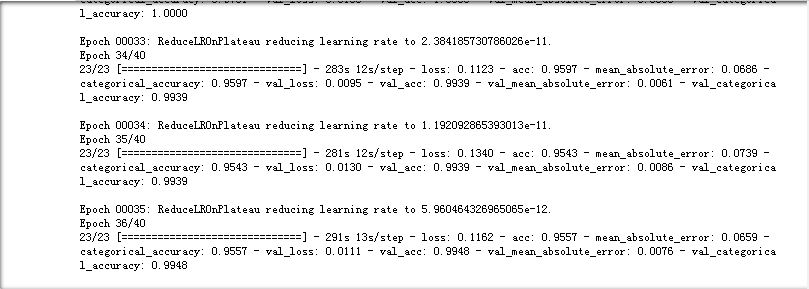












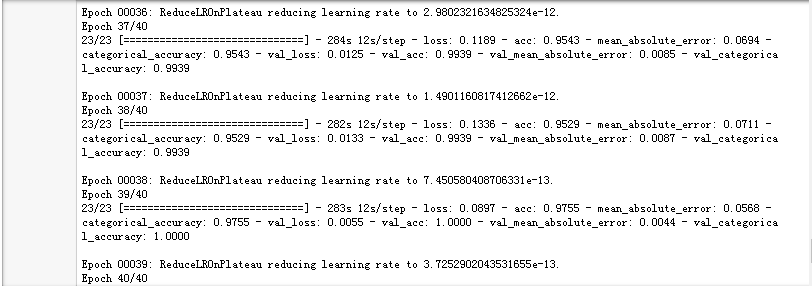


图17 训练结果图

在多次调节参数训练后，任意选取了其中一次的准确率和损失函数的更新曲线，曲线如图18、19所示。

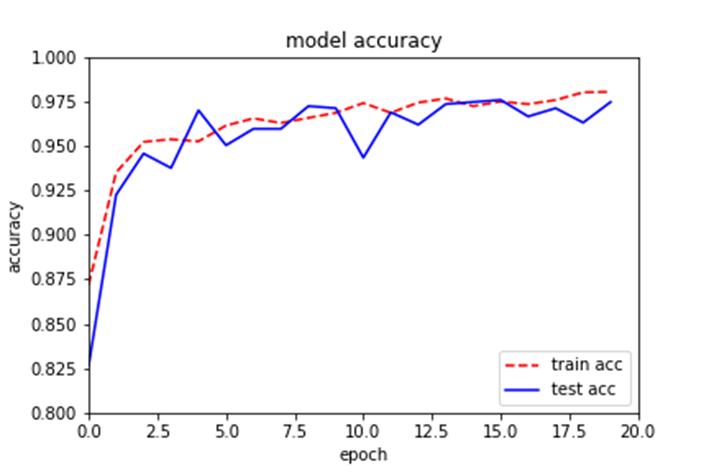


图18 准确率更新曲线

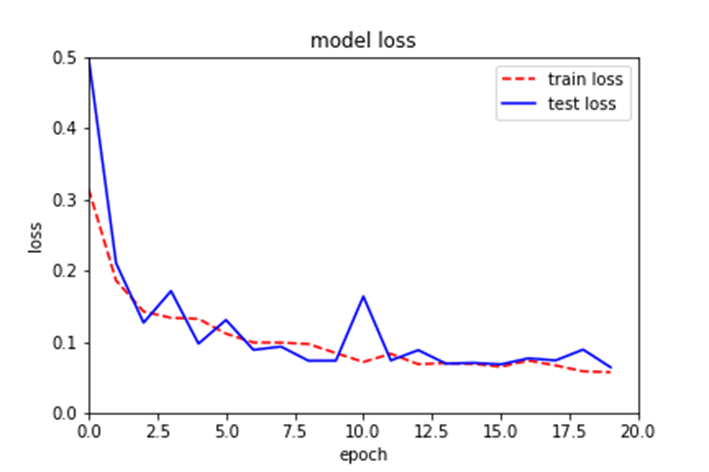
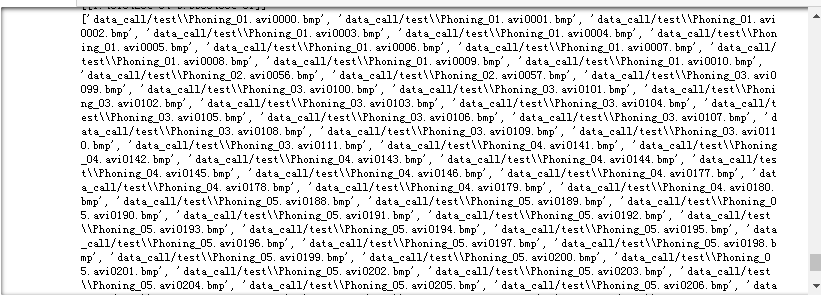


图19 损失函数曲线

使用该训练好的模型对test进行预测，输出文件中为calling的图片，结果如图20.



几次预测的准确率在97%左右，网络效果良好。

# 4 感想和展望

首先，非常感谢老师和助教在半年内的付出。杨老师致力于让我们从书本知识中得到感悟和创新，经常在课上进行井喷式思维挖掘，这一点我感触很深，助教师兄传授了很多debug经验和知识理论，很是受用。

对于本作业，难度较大，本人比较愚钝，大概做了不到一个月才完成，前面几天和同学讨论过思路，查阅过文献，虽然没有太受用，但是还是启发了一些小思路。

希望能通过此类的训练提升自己的思维能力和独立动手能力，在今后的学习工作中多多做一个有心人，尽量在能完美简练解决问题的基础上形成自己的思考，而不是人云亦云。

谢谢阅读。