西北工业大学

“算法设计综合实验”报告

报告题目：

基于视觉的人员检测识别

班级	学号	姓名	承担的主要工作
09061702	2017302230	陈湘媚	网络搭建（人员检测）、界面设计、代码融合
09061702	2017302229	杨一哲	资料收集、摄像头标定及视差图计算

2019 年 9 月 2 日

一、应用需求：

项目背景：

伴随人工智能及的发展，人员识别技术作为计算机视觉技术的一部分逐步迈入了成熟。越来越多的领域用到了人员识别及定位技术，如无人驾驶、引领导航、智能避障、安全防控等，这也就要求我们能够在实时的情况下快速对人员进行检测定位。

现阶段人员检测方法众多，常用的方法有图像法、SVM 分类器法、机器学习法等，其中最为常用的是机器学习法。而随着机器学习的不断成熟发展，各大神经网络也在发挥着重大的作用，常见的 VGGNet、GoogleNet、ResNet 等网络都能够较好地实现人员的识别，并且在 GPU 上的运行速度也能够满足实时性的要求。但并非所有的设备都具有 GPU，大多手机端或在只有 CPU 的设备上还存在运行速度的问题，因此需要优化加速网络。而设计轻量级的神经网络就能够有效地解决这个问题。

对于人员的定位问题，大多使用摄像头来实现。目前主流摄像头主要分为三种，单目、双目、深度摄像头。用单目摄像头进行测距具有低成本和计算快的优点，但缺点在于需要依靠大量的数据进行计算。深度摄像头利用深度传感器检测深度，测量结果较准，但市面上的深度摄像头通常价格较为昂贵。而双目摄像头通过计算视差图来进行测距，精度较准，价格相对较低，有一定的性价比。

因此，本次实验将通过搭建轻量级神经网络 MobileNet，使用 SSD 目标检测算法检测目标人员，并利用双目摄像头实现人员的定位功能。

二、设计框架：

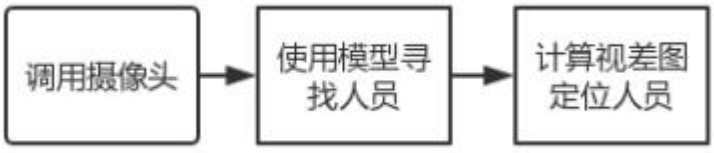


图 1 双目摄像头寻找及定位人员流程

如图 1,是利用双目摄像头实时寻找及定位人员的流程，该软件使用语言为 python，且在 linux、windows 系统下均可实现。流程主要分为调用摄像头、使

用模型寻找人员、计算视差图并定位人员三大块，下面将对三个部分做简要说明。

调用摄像头：

为能够获取到摄像头的图像信息，我们使用 python-opencv 库，opencv 在 python 下的图像是以 numpy 的 array 来进行表示的。获取到的图像的大小为 640×480，随后我们通过调用 Segmentation(Frame)函数进行图像的分割从而得到左右图像，大小为 320×240

使用模型寻找人员：

在寻找人员这一块，我们借用机器学习训练的模型来实现。调用模型进行人员识别的功能放在了 Detect 类中，该类包含了两个类成员变量和两个类成员函数，这两个类成员函数主要用于加载网络及模型（Init_Net）和检测（detect）。在 Init_Net 函数中，涉及到另外两个重要函数，一是加载网络的函数：create_mobilenetv2_ssd_lite；一是进行人员预测的函数：create_mobilenetv2_ssd_lite_predictor。其中较为重要的是 create_mobilenetv2_ssd_lite 函数，该函数实现了 mobilenet-ssd 网络结构的搭建。在该函数下，又主要调用了两个类，一为 MobileNetV3 类，该类搭建了 MobileNetV3 的网络结构；第二为 SSD 类，该类主要实现了 SSD 目标检测算法，而 create_mobilenetv2_ssd_lite 函数实现了两者的融合，并传递出最终的网络结构（mobilenet-ssd）。这一部分的函数调用关系如图 2 所示：

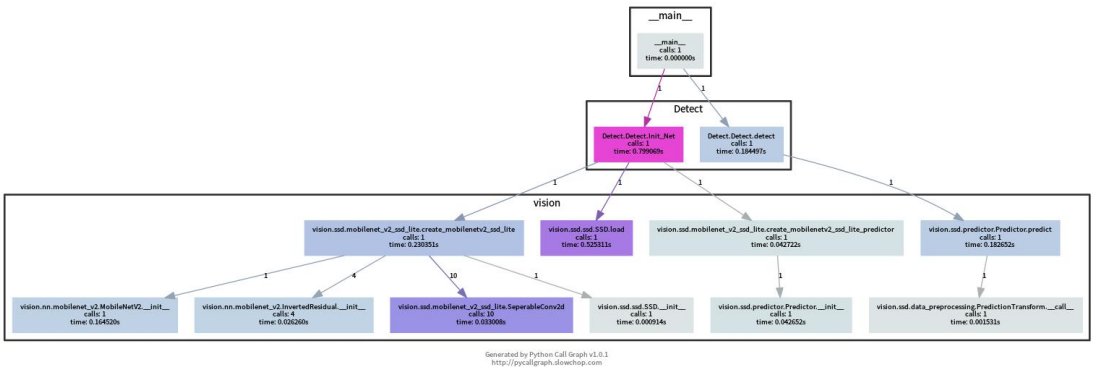


图 2 人员检测函数调用关系

初始化结束后，将把摄像头分割出来的图像传入 detect 函数中，通过预测模型的检测，筛选出结果，并进行结果的返回，这里结果被存入一个二维列表（list）里，每一个子列表包含人员在 RGB 图像中的坐标信息。

计算视差图定位人员：

最后一部分，是通过计算双目摄像头的视差图来计算深度信息，这里我们建立了 SGBM 类，该类下主要函数为 Coordination()，传入左右图像以及得到的人员坐标二维列表，最后依照传入的坐标顺序分别输出每一个识别到的人距离摄像头的实际距离。

除此之外，我们的界面使用了 PyQt5 进行设计。

三、主要算法思想、实现框架：

我们将从识别并定位人员、界面设计两方面进行详细阐述：

1、识别并定位人员：

在该部分我们将说明实验涉及到的 MobileNet-SSD 算法和 SGBM 算法。

1.1、MobileNet-SSD 算法

在人员寻找方面，我们使用了神经网络技术，并搭建了轻量级的神经网络，来使得无 GPU 设备也能实时快速且准确地识别检测人员。

目前为止，现有的轻量级神经网络有 MobileNet、SqueezeNet、ShuttleNet 等，这里我们主要对 MobileNet 进行研究。

通常来说，标准的卷积过程是使用卷积核进行卷积的时候，对应图像区域中的所有通道均被同时考虑，但 MobileNet 中使用了一种名为深度可分离卷积的新思路，其认为，对于不同的输入通道可以采取不同的卷积核进行卷积，这就意味着它将标准卷积分成了一个深度卷积和一个点卷积。计算量如下所示：

$$\frac{depthwise + pointwise}{conv} = \frac{H \times W \times C \times 3 \times 3 + H \times W \times C \times k}{H \times W \times C \times k \times 3 \times 3} = \frac{1}{k} + \frac{1}{3 \times 3}$$

从上式可以看出，MobileNet 有效地减少了网络需要的参数，极大地降低了计算量，并且从作者给出的训练结果上来看准确率没有出现降低严重的状况。

MobileNetV2 是对 V1 的改进，V1 结构虽然简单，但性价比不高，MobileNetV2 在 V1 的基础上，使用逆残差结构，与原始的残差结构不同的是，逆残差结构会先通过 1*1 卷积提升通道数，再通过 Depthwise 的 3*3 空间卷积，最后通过 1×1 卷积降低纬度。除此之外，它还提出将最后一层的 ReLU6 激活函数去掉，直接线性输出。

最后，MobileNetV3 作为新提出来的一种网络结构，结合了 V1、V2 版本的优势，又做出了改进。作者采用互补搜索技术，一种为整体式的结构搜索，一种为局部式的结构搜索。并且进行了网络结构的改进，该网络在平均池化前的层移除并用 1*1 卷积来计算特征图，这样提速的同时也减少了操作数。在 MobileNetV3 中，作者使用了一种新的激活函数改进网络精度，如下所示。

$$\text{h-swish}[x] = x\frac{\text{ReLU6}(x+3)}{6}$$

在本次实验中，我们主要使用了 MobileNetV2 和 V3 进行测试。

为了能够检测并定位人员，我们还使用了目标检测算法。目前主要的目标检测算法有 Fast-RCNN、SSD、YOLO 等，这里我们使用了 SSD 算法。SSD 算法利用不同的卷积层的 feature map 进行综合达到预测目标类别和目标框（bounding box）的效果。该算法的实现主要是将网络结构的后两个全连接层改成卷积层，并增加 4 个卷积层来构造网络结构。采用两个不同的 3*3 的卷积核卷积 5 种的卷积层，一个输出分类用的 confidence，一个输出回归用的 localization，最后将三个计算结果分别合并然后传给 loss 层。

所以，最终我们的人员检测采用了 MobileNet 网络结构与 SSD 目标检测算法相结合的方式来实现。对于 MobileNetV2-SSD 来说，我们舍弃最后一层全连接层，并加上 4 个卷积层得到最终的网络结构，不过在得到 confidence 和 localization 方面，我们使用了两个 3*3 的卷积核卷积 6 种卷积层来进行输出。

而对于 MobileNetV3-SSD，首先 MobileNetV3 有两种结构，一个是 MobileNetV3-Large，一个是 MobileNetV3-Small，对于两者我们都进行算法的融合改进，这里与 MobileNetV2 的结构相似，就不再叙述。

在这一部分，我们使用了 Pytorch 开源框架来实现了网络的搭建和算法的实现。以下为对应的主要的类和函数的功能叙述：

MobileNetV2(nn.Module)：

搭建 MobileNetV2 网络结构

MobileNetV3(nn.Module)：

搭建 MobileNetV3 网络结构，包括 MobileNetV3-Large 和 MobileNetV3-Small

create_mobilenetv2_ssd_lite(num_classes, width_mult=1.0, use_batch_norm=True, onnx_compatible=False, is_test=False):

创建 MobileNetV2-SSD 网络结构并获取 confidence 和 localization

create_mobilenetv3_ssd_lite(moudle, num_classes, width_mult=1.0, use_batch_norm=True, onnx_compatible=False, is_test=False):

创建 MobileNetV3-SSD 网络结构并获取 confidence 和 localization

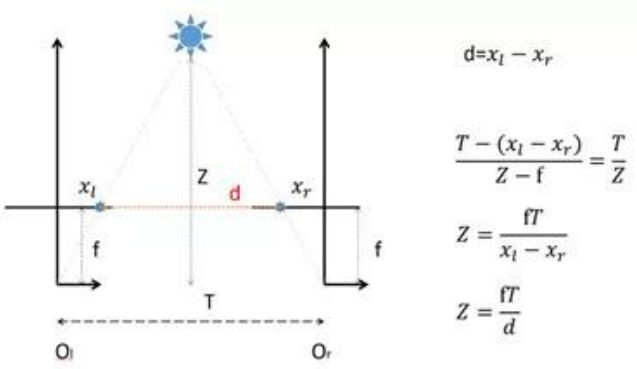
SSD(nn.Module):

SSD 算法的实现

在训练模型中，我们对数据集进行了数据的增广并采用了 SGD 的优化方法。模型训练完毕后，对模型的评估算法放在了 eval_ssd.py 文件中。

1.2、SGBM 算法

以摄像头的三维坐标为准，假设目标物体位于坐标系的(x0,y0,z0)一点，那么，可以得到如下表达式：



其中，f 为摄像头的焦距，T 为左右相机之间的距离，xl 为该点与坐标原点的连线与画面交点的 x 距离，xr、yl、yr 同理。因此，如果得到了相机内部的参数，将左右图片进行校正使之平行，那么就可以通过计算视差图来获取到该点在世界坐标系中的实际位置。

为获得视差图，我们需要进行立体匹配。立体匹配算法有多种，在这里我们选取了半全局块匹配算法即 SGBM 算法，并利用 Opencv 进行实现。

SGBM 算法获取视差图会经过如下操作：首先进行图像归一化，减少亮度差别，增加纹理；随后，滑动 sad 窗口，沿着水平线进行匹配搜索，通过左图的

特征在右图中找到最佳匹配，并进行过滤，去除掉坏的匹配点，最终输出视差图。

这一块的内容，我们放在了 SGBM 类中进行了实现。其类下，Init_SGBM()用于加载 SGBM 的参数，Coordination()函数通过计算传来的左右视图以及框选出来的人的坐标来计算该人相对于摄像头的距离。

2、界面设计：

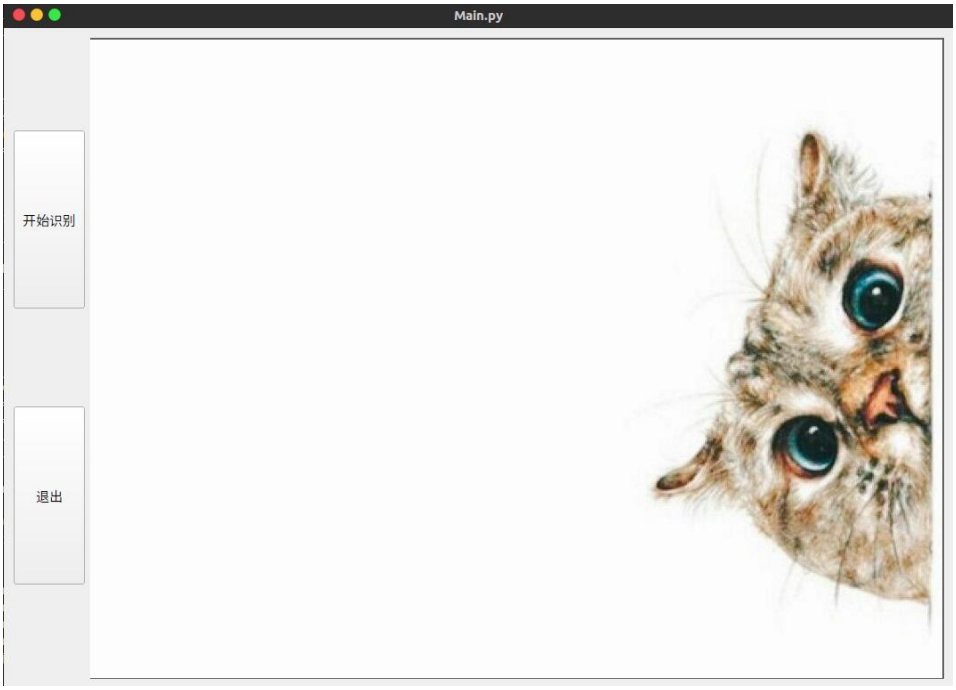
对于界面我们使用了 PyQt5 进行设计，界面将有两个按钮，一个是“开始识别”，一个是“退出”。

点击“开始识别”后会弹出摄像头的左右视图、计算出来的视差图以及目标识别图，计算出来的距离信息会直接显示在图片上。此外点击“开始识别”之后该按钮会变成“结束识别”，此时可以选择继续或者退出。

四、测试：

鉴于本实验需要使用双目摄像头，因此最后放在一台无 GPU 的电脑上进行测试：

界面打开如下：



点击开始识别会弹出摄像头的左右视图、计算出来的视差图以及目标框选图：

距离信息目前我们是直接显示在图片上了

五、性能分析及结果：

用模型检测测试集的数据，并计算出平均预测结果，为 0.669：

```
Average Precision Across Class-Person:0.6690656284914068
```

而在无 GPU 电脑上运行整个程序，视频流可达 7-8fps/帧数

```
The number of person: 1
Detect time:0.001268625259399414
[[tensor(8.2723), tensor(43.1695), tensor(301.3262), tensor(237.2798)]]
Coordination time:0.11838936805725098
Estimated frames per second : 7.296370518815409
```

六、总结：

亮点：

1、采用轻量级神经网络 MobileNetV2 以及 MobileNetV3 与目标检测算法 SSD 相结合的方法来进行人员的识别检测，网上大多关于 MobileNet-SSD 的实现都利用的 Caffe 或 Tensorflow，并且 MobileNetV3 与 SSD 的融合少有实现，而本实验中使用 Pytorch 框架，利用 Pytorch 的易读易写的优势搭建网络，也实现了 MobileNetV3-SSD 的搭建。

2、使用双目摄像头计算视差来测量人员的距离，从而实现定位的功能。相比于单目测距，双目带来的视差可以更为准确地计算出具体位置信息，而相较于深度摄像头测距，双目的价格具有极大的优势。

实验中遇到的问题及解决：

本次实验主要遇到了以下几个问题：

1、视差图噪点多，测量距离浮动大且不准：

解决方案：视差图的好坏取决于摄像头的标定，为获得更为准确的相机内部参数，我们后来拍摄多张多角度、多尺寸的标定图，并通过不断地试验发现，先单目标定再双目标定获取到的内部参数会比直接进行双目标定获取的参数精确很多。通过多次尝试终于获取到了较好的视差图，测量结果也准确许多。

2、人员识别效果差，精度低：

解决方案：人员识别的效果取决于模型的好坏，刚开始给的学习率太高，导致 loss 值波动很大，我们尝试了两种方式，一个是每到一个设定的 epoch 后就使学习率缩小十倍，二是利用 cos 曲线降低学习率，该方式的公式如下所示：

$$\eta_t = \eta_{\min} + \frac{1}{2}\left(\eta_{\max} - \eta_{\min}\right)\left(1 + \cos\left(\frac{T_{cur}}{T_{\max}}\pi\right)\right)$$

通过试验发现，后者训练得到的模型更利于识别

3、网络搭建失败：

解决方案：网络搭建的问题是一开始最容易出现也是较为麻烦的问题，虽然按照官方给的网络结构进行搭建了，但测试的时候总是显示通道数不对或者尺寸不对，该问题的解决也比较容易，就是耐下心来仔细检查搭的网络，弄懂每一层做了些什么。

反思与改进：

本次实验虽然达到了一开始想要的目的，但是总的来说也有没有做好或者说需要改良的地方。

第一，本次实验采用 MobileNet 神经网络，是因为看了官方给的结果，尤其是新出的 MobileNetV3 的结果，觉得是个不错的选择，而忽略了对其他轻量级网络的尝试。

第二，SSD 目标检测算法虽然相较于 Fast-RCNN 等速度快很多，但并非最优选择。YOLO 算法相较于 SSD 简洁一些，甚至识别效果更优。而 MobileNet-YOLO 也有人提出过，本次试验没有尝试过 MobileNet-YOLO 的搭建，如果后续有时间和机会，可以好好做一番。

第三，这次因为使用了双目摄像头进行定位，所以没有在手机端上进行尝试。