廣東工業大學

本科毕业设计（论文）

基于 python 的深度学习 tensorflow 立体视觉实现

学　　院　物理与光电工程学院

专　　业　光电信息科学与工程

（光电方向）

年级班别　2014 级光电（1）班

学　　号　3114008585

学生姓名　余梓煜

指导教师　徐胜

2018 年 5 月

摘　要

随着现代化科技的快速发展，生活中人工智能的成果比比皆是，例如：人脸识别技术、智能语言识别、智能聊天机器人、智能交通指挥、图像分类技术、3D 重建技术等；而在人工智能领域，计算机视觉领域的研究已经逐步成熟，其中立体视觉作为计算机视觉中最重要的一部分，已经在智能交通、人脸识别、信息采集、图像检索中发挥着极其重要的作用。

本文的立体视觉实现由目标检测和目标测距组成。立体视觉的实现，往往离不开目标的检测,而有很多传统的目标检测算法是基于手工设计的特征和小训练样本的，经常受到光照、遮挡等外界因素的影响，进而影响目标检测模型的性能，而深度学习在抗外界干扰的鲁棒性很好,因此本模型采用检测性能更好的深度学习算法搭建，而目标测距主要用到双目视觉实现。

关键词：立体视觉，目标检测，目标测距，深度学习

Abstract

With the rapid development of modern science and technology, there are many achievements in artificial intelligence in life, such as face recognition technology, intelligent language recognition, intelligent chat robot, intelligent traffic command, image classification technology, 3D reconstruction technology, and so on, in the field of artificial intelligence, the research of computer vision field has been gradually mature. As the most important part of computer vision, stereovision has played an important role in intelligent traffic, face recognition, information collection and image retrieval.

The realization of stereo vision is composed of target detection and target ranging. The realization of stereo vision is often inseparable from the detection of the target, and many traditional target detection algorithms are based on the manual design features and small training samples, often affected by the external factors such as illumination and occlusion, and then influence the performance of the target detection model, while the depth learning is robust to the external interference. Therefore, this model builds a deep learning algorithm with better detection performance； The target location is mainly used to realize the binocular vision.

Keywords: Stereo Vision, Object Detection, Object Location, Deep learning

目　　录

1 绪论

1.1 研究背景及意义

在人工智能领域，机器视觉的研究基本出于前沿地位，计算机视觉是一门让机器学会“看”世界的学科，通过各种各样的硬件传感器来代替人眼的作为输入，并对输入的图像做进一步处理，使得其更加适合人眼观察或者其他仪器检测，而立体视觉是机器视觉中的主要领域,立体视觉主要的方向有图片的 3D 重建、目标的检测、目标的识别、目标的测距等；

随着计算机技术的飞速发展，深度学习（神经网络）在机器视觉中的运用已越发成熟和热门，人们希望深度学习能够完全代替人们工作，因此在机器视觉领域不断寻求突破，在 2016 年 03 月，围棋人工智能机器“AlphaGo”击败了韩国职业围棋选手李在石，“AlphaGo”为何能够击败职业围棋高手呢？其实答案是毋庸置疑的，“AlphaGo”背后是由成千上万个分布式的 GPU 组成，运行速度非常之快，因为深度学习的神经网络运行需要建立在高速处理的处理集群中，时效性才能在所要求范围内，这一事件说明了人工智能、深度学习的成熟技术正越来越多应用到生活当中，也相信在不久的将来，我们身边会随处可见，具有智能大脑的机器。

总而言之，机器视觉的研究已经越发成熟，而立体视觉的重建，包括目标检测和目标测距技术也在进一步革新中，所以本文是基于 python 和 tensorflow 的目标检测与目标测距来对环境中的目标进行相关的技术实现。

1.2 国内研究主要方向

1.2.1 国内外目标检测方向相关研究

1.2.1.1 基于传统人工特征的目标检测

目标检测是计算机视觉研究的大方向，所以国内外有很多学者在研究这方面的技术，传统的人工特征目标检测包括两方面：目标特征的提取以及分类，目前比较主流的人工目标特征提取的方法：LBP 特征，HOG 特征等

LBP（Local Binary Pattern）：主要用于图像纹理特征的提取，LBP 根据像素周边的 8 个临近像素的灰度值，和中心灰度值进行比较，比较的结果是得出八位的有

效编码，然后再根据这些编码来进行相关的分类；

HOG（Histogram of Oriented Gradient）：主要是使用基于 HOG 的单独模板的滤波器，采用滑动窗口的方法，将滤波器应用到的图像的所有可能的位置和尺度。此类滤波器又是分类器，可以用来判断在指定位置和尺度中是否有目标类别的存在。

1.2.1.2 基于深度学习的目标检测

实际上，传统的人工检测效果并不尽人意，因为传统的人工识别，过分依赖人为定义特征，渐渐无法适合如今大数据优势，往往只适合某类目标检测，更换数据集之后，达不到预期效果；从 2006 年以后，深度学习在计算机视觉、人工智能、机器学习等领域就受到了广泛的关注，并且取得了不错的成绩，例如：谷歌公司利用深度学习算法实现了精准度达到 99%的目标识别，阿里巴巴利用深度学习来进行智能化交通中车牌号码的识别等等；种种实现成果表明，深度学习已经越发成熟，越来越多科学家在深度学习模型领域不断努力。

1.2.2 国内外目标测距方向的相关研究

1.2.2.1 基于单目视觉的目标测距（传统数学原理）

国内外在于单目视觉测距方面主要利用的方法主要有利用小孔成像原理、相似三角形计算相机到一个已知物体或者目标的距离。

1.2.2.2 基于双目视觉的目标测距

双目立体视觉测距主要是利用三角形原理，具体实现原理如下图所示：物体或目标的某点 P 在摄像机 1 和摄像机 2 上的成像点分别为 P1 和 P2 图像上成像点 P1 和 P2 相对于坐标原点 O1 和 O2（O1 和 O2 分别是左、右相机透镜光轴与图像平面的交点）的距离分别为 x1 和 x2，P 点再左、右图像平面上成像点的位置差（x1-x2）称为视差，由三角形相似原理，可得：

$$\frac{d-(x_1-x_2)}{Z-f}=\frac{B}{Z}\longrightarrow Z=\frac{fd}{x_1-x_2}$$

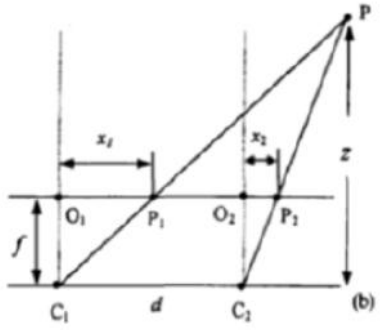

图 1　双目立体视觉测量原理

图 1.1 双目立体视觉测量原理

1.3 本文研究内容以及章节安排

本文开篇介绍了立体视觉、目标检测、目标测距在机器视觉中的重要性，以及国内外主要的目标检测和目标测距相关交前沿的科学研究，并且对相关一些比较热门的算法进行较为详细的介绍，接下来的章节安排如下所示：

第一章主要介绍立体视觉、目标检测、以及目标测距的研究背景和意义，以及国内外相关领域的研究进展，并对相关算法进行较为详尽的介绍；

第二章主要介绍本文所涉及相关技术的发展历史和基础原理，并对本文所用到的相关原理进行更加详细的介绍和拓展；

第三章主要介绍基于深度学习的目标检测算法的种类，以及对本文所用到的算法 YOLO 算法的优点和架构进行相应的分析；

第四章主要介绍基于相似三角形原理的单目目标识别和测距的相关原理，以及对核心代码的设计思想进行相应的分析；

第五章主要是对本文所有工作的总结，对于本文中海存在的不足之处进行合理性的分析，展望以后的研究内容与工作。

2 相关工作基础原理介绍

2.1 立体视觉的历及发展

立体视觉是计算机视觉研究中的重点领域，在 20 世纪的 60 年代中期开始，美国麻省理工大学的 Robert 教授把二维图形分析扩展到三位景物分析，标志着计算机立体视觉技术的诞生，并且在随后的几年中，迅速成为一门崭新的学科，尤其在 20 世纪的 70 年代中期，Marr 教授等首创的视觉计算理论对于立体视觉的发展起到了至关重要的作用，现阶段，已经形成了从图像获取到最终景物可视表重建的比较完整的体系；立体视觉主要包括单目立体视觉和双目立体视觉；立体视觉具体的实现步骤，如下所示：

第一步：图像的获取：二维图像始终是立体视觉信息的来源，一般通过数码相机、摄像机或者扫描仪等输入设备实现，作为需要进行三维重建的物体的对象；

第二步：摄像机的标定：主要是对摄像机位置以及内外属性进行确定，达到确定物体点和它的像点之间存在的对应关系；

第三步：特征的提取：获得图像数据中最能够突出物体属性的相关信息，实质上就是对同 一物点不同角度下图像中存在的特定对应关系进行确定；

第四步：立体匹配：对所选特征进行计算建立起特征间对应关系的过程，是立体视觉中摄像机图像最为关键的一步，其基本原理就是选择出合理的基元，借助一定的计算获得两幅图像中基元之间的对应关系；

第五步：三维重建：对场景中的物体点实施三维重建，即是对物体表面点所蕴含的三维信息进行恢复；

第六步：目标测距：已知图像点（x，y）和视差值 d，可通过物理公式计算出物体深度以及目标距离。

2.2 目标检测算法介绍

目标检测近年来已经取得了一些很重要的进展，主流的算法主要分为两个类型：

1、one-stage 方法，如 YOLO（You Only Look Once）和 SSD 等，其中主要的思路是均匀地在图片的不同位置进行密集抽样，抽样时可以采用不同尺度和长宽比，然后利用 CNN 提取特征后直接进行分类和回归，整个过程只需要一步，所以优势就是速度快，但是均匀的密度采样的一个重要的缺点是训练比较困难，其不均衡导致模型准确度稍低。

2、two-stage 方法，如 R - CNN、R - FCN、SPP - Net、Fast - R -CNN 等算法，其主要的思想是先通过启发式方法（selective search）或者 CNN 网络（RNN）产生一系列稀疏的候选框，然后对这些候选框进行分类和回归，two - stage 方法的优势就是准确度高，但是运行速度会慢一些。

2.3 目标测距算法介绍

2.3.1 基于单目视觉的目标测距（传统数学原理）

具体实现原理如下所示：

假设有一宽度为 W 的目标或者物体，将该目标或者物体放在距离相机为 D 的位置，我们用相机对物体进行拍照，并计算物体的像素宽度 P，这样我们就可以得出相机的焦距公式：F=（P*D）/W

当我们将移动或者原理物体的时候，可以利用相似三角形计算出目标或者物体到相机之间的距离：D’=（W*F）/P

我们利用的方法是使用相似三角形计算相机到一个已知物体或者目标的距离。

2.3.2 基于双目视觉的目标测距

双目立体视觉测距主要是利用三角形原理，具体实现原理如下图所示：物体或目标的某点 P 在摄像机 1 和摄像机 2 上的成像点分别为 P1 和 P2 图像上成像点 P1 和 P2 相对于坐标原点 O1 和 O2（O1 和 O2 分别是左、右相机透镜光轴与图像平面的交点）的距离分别为 x1 和 x2，P 点再左、右图像平面上成像点的位置差（x1-x2）称为视差，由三角形相似原理，可得：

$$\frac{d-(x_1-x_2)}{Z-f}=\frac{B}{Z}\longrightarrow Z=\frac{fd}{x_1-x_2}$$

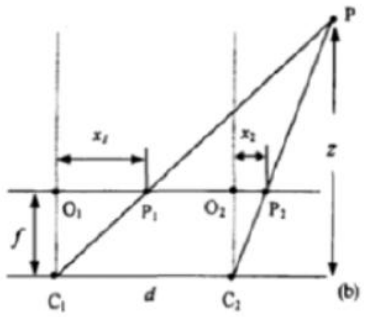

图 1　双目立体视觉测量原理

图 2.1 双目立体视觉测量原理

2.4 实验所用操作环境和工具

2.4.1 操作环境：Ubuntu 操作系统、python3.5、Tensorflow

2.4.1.1 Ubuntu 操作系统：Ubuntu（乌班图）是一个以桌面应用为主的 Linux 操作系统，其名称来自非洲南部祖鲁语或豪萨语的“Ubuntu”一词，意思是“人性”、“我的存在是因为人家的存在”，是非洲传统的一种价值观，类似华人社会的“仁爱”思想。Ubuntu 基于 Debian 发行版和 GNOME 桌面环境。而从 11.04 版开始，Ubuntu 发行版本放弃了 Gnome 桌面环境，改为 Unity，与 Debian 的不同在于它每 6 个月会发布一个新版区力量，用户可以方便地从社区获得帮助。

Ubuntu 基于 Linux 的免费开源桌面 PC 操作系统，十分契合英特尔的超极本定位，支持 X86、X64 位和 PPC 架构。

2.4.1.2 python：python 编程语言是一种面向对象的解释型计算机程序设计语言，由荷兰人 Guido van Rossum 于 1989 年发明，第一个公开发行版发行于 1991 年；python 是纯粹的自由软件，源代码和解释器 CPython 遵循 GPL（GNU General Public License）协议。Python 语法简洁清晰，特色之一是强制用空白符（white space）作为语句的缩进；python 具有丰富和强大的库，它常被昵称为胶水语言，能够把用其他语言制作的各种模板（尤其是 C++/C#）很轻松地联结在一起，常见的一种应用情景，使用 python 快速生成程序的原型（有时候甚至是程序的最终界面），

然后对其中有特别要求的部分，用更合适的语言改写，比如 3D 游戏中的图像宣传模块，性能要求特别高，就可以用 C/C++重写，而后封装为 python 可以调用的扩展类库。需要注意的是在你需要使用扩展类库的时候需要考虑平台的问题，某些不提供跨平台的实现；在 2018 年 05 月的编程语言排行榜中，python 超过 java 高居第一位。

2.4.2 实验工具：Tensorflow、Pycharm、Jupyter Notebook

2.4.1.3 Tensorflow：Tensorflow 是谷歌基于 DistBelief 进行研发的第二代人工智能学习系统，其命名来源于本身的运行原理：Tensor（张量）意味着 N 维数组，Flow（流）意味着基于数据流图的计算，Tensorflow 为张量从流图的一端流向另一端计算的过程。Tensorflow 是将复杂的数据结构传输至人工智能神经网络中进行数据分析和处理过程的系统；Tensorflow 可被用于语言识别和图像识别等多项机器深度学习领域，对 2011 年开发的深度学习基础架构 DistBelief 进行了各方面的改进，它可在小到一部智能手机、大到数千台数据中心服务器的各种设备上进行。Tensorflow 是完全开源的，任何感兴趣的人都可以使用。

2.4.1.4 Pycharm：PyCharm 是一种 Python IDE，带有一整套可以帮助用户在使用 Python 语言开发时提高其效率的工具，比如调试、语法高亮、Project 管理、代码跳转、智能提示、自动完成、单元测试、版本控制。此外，该 IDE 提供了一些高级功能，以用于支持 Django 框架下的专业 Web 开发。PyCharm 是由 JetBrains 打造的一款 Python IDE，VS2010 的重构插件 Resharper 就是出自 JetBrains 之手。

同时支持 Google App Engine，PyCharm 支持 IronPython。这些功能在先进代码分析程序的支持下，使 PyCharm 成为 Python 专业开发人员和刚起步人员使用的有力工具。

首先，PyCharm 用于一般 IDE 具备的功能，比如， 调试、语法高亮、Project 管理、代码跳转、智能提示、自动完成、单元测试、版本控制

另外，PyCharm 还提供了一些很好的功能用于 Django 开发，同时支持 Google App Engine，更酷的是，PyCharm 支持 IronPython。

2.4.1.5 Jupyter Notebook：Jupyter Notebook(此前被称为 IPython notebook)是一个交互式笔记本，支持运行 40 多种编程语言。

Jupyter Notebook 的本质是一个 Web 应用程序，便于创建和共享文学化程序文档，支持实时代码，数学方程，可视化和 markdown。 用途包括:数据清理和转换，数值模拟，统计建模，机器

学习等等 。

用户可以通过电子邮件，Dropbox，GitHub 和 Jupyter Notebook Viewer，将 Jupyter Notebook 分享给其他人。

在 Jupyter Notebook 中，代码可以实时的生成图像，视频，LaTeX 和 JavaScript。

为什么使用 Jupyter

数据挖掘领域中最热门的比赛 Kaggle 里的资料都是 Jupyter 格式 。

Jupyter 包含以下组件:

1、Jupyter Notebook 和 Notebook 文件格式

2、Jupyter Qt 控制台

3、内核消息协议 (kernel messaging protocol)

4、许多其他组件

Jupyter Notebook 与 IPython 终端 共享同一个内核[1] 。

内核进程可以同时连接到多个前端。 在这种情况下，不同的前端访问的是同一个变量[1] 。

这个设计可以满足以下两种需求:

1、相同内核不同前端，用以支持，快速开发新的前端

2、相同前端不同内核，用以支持，新的开发语言

Jupyter Notebook 界面由以下部分组成：

1. notebook 的名称

2. 主工具栏，提供了保存、导出、重载 notebook，以及重启内核等选项

3. 快捷键

4. notebook 主要区域，包含了 notebook 的内容编辑区

2.4 本章小结

本章主要介绍了立体视觉的发展史，基于深度学习的目标检测算法种类，以及单目、双目视觉测距的原理，从相关原理知识可以看出，立体视觉重建、目标检测、目标测距技术已经越发成熟，而本章节只是就现阶段在立体视觉重建、目标检测、目标测距的热门研究进行相应的总结，但由于本人才识学浅，无法将国内外所有立体视觉相关热门前沿研究的写入其中，敬请老师谅解。

3 基于深度学习算法目标检测

3.1 YOLO 算法目标检测原理

3.1.1 YOLO 算法核心思想定义

3.1.1.1 网络定义

YOLO 检测网络包括 24 个卷积层和 2 个全连接层，如下图所示：

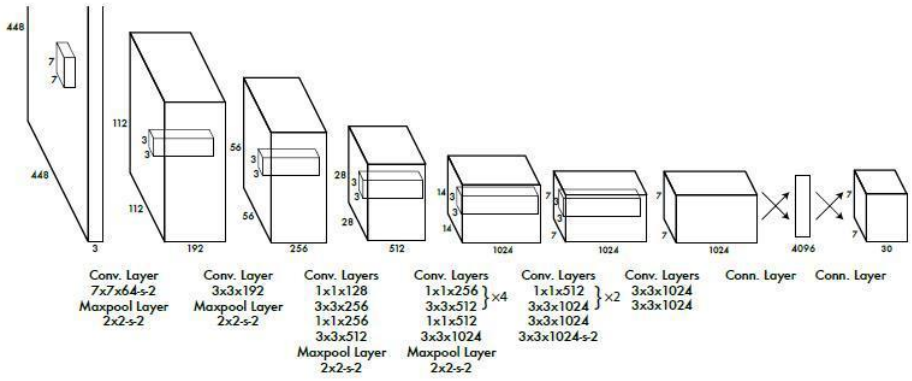

Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

图 3.1 YOLO 算法结构图

其中，卷积层用来提取图像特征，全连接层用来预测图像位置和类别概率值。YOLO 网络借鉴了 Google Net 网络网络结构，不同的是，YOLO 网络未使用 inception Module，而是采用 1x1 的卷积层 3x3 卷积层简单替代。

3.1.1.2 LOSS 函数定义

YOLO 使用均方和误差作为 loss 函数来优化模型参数，即网络输出的 S*S*(B*5 + C)维向量与真实图像的对应 S*S*(B*5 + C)维向量的均方和误差。如下式所示。其中 *iouError* 和 *classError* 分别代表预测数据与标定数据之间的坐标误差、IOU 误差和分类误差。

$$loss = \sum_{i=0}^{S^2} coordError + iouError + classError$$

图 3.2 LOSS 函数

3.1.2 YOLO 算法创新点

YOLO 将物体检测作为回归问题求解。基于一个单独的 end-to-end 网络，完成从原始图像的输入到物体位置和类别的输出。从网络设计上，YOLO 与 R- CNN、FAST　R- CNN 及 FASTER R- CNN 的区别如下：

[1] YOLO 训练和检测均是在一个单独网络中进行。YOLO 没有显示地求取 region proposal 的过程。而 R- CNN、FAST　R- CNN 采用分离的模块（独立于网络之外的 selective search 方法）求取候选框（可能会包含物体的矩形区域），训练过程因此也是分成多个模块进行。FASTER R- CNN 使用 RPN（region proposal network）卷积网络替代 R- CNN/FAST　R- CNN 的 selective search 模块，将 RPN 集成到 FAST R- CNN 检测网络中，得到一个统一的检测网络。尽管 RPN 与 FAST　R- CNN 共享卷积层，但是在模型训练过程中，需要反复训练 RPN 网络和 FAST　R- CNN 网络（注意这两个网络核心卷积层是参数共享的）。

[2]YOLO 将物体检测作为一个回归问题进行求解，输入图像经过一次 inference，便能得到图像中所有物体的位置和其所属类别及相应的置信概率。而 R- CNN/FAST R- CNN/FASTER R- CNN 将检测结果分为两部分求解：物体类别（分类问题），物体位置即 bounding box（回归问题）。

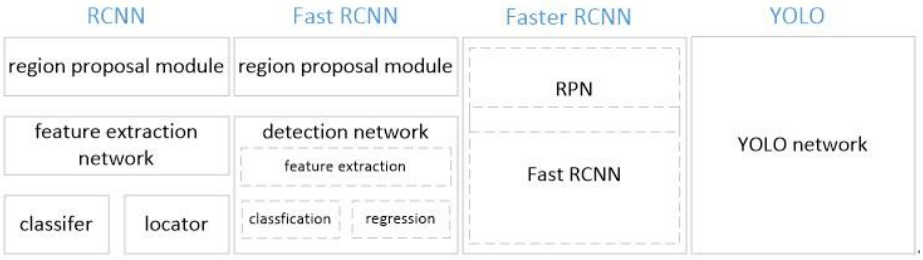

图 3.3 YOLO 与 RCNN 之间比较

3.1.3 YOLO 具有以下特点

1、快，YOLO 将物体检测作为回归问题进行求解，整个检测网络 pipeline 简单。在 titan x GPU 上，在保证检测准确率的前提下（63.4% mAP，VOC 2007 test set），可

以达到 45fps 的检测速度；

2、背景误检率低。YOLO 在训练和推理过程中能‘看到’整张图像的整体信息，而基于 region proposal 的物体检测方法（如 R- CNN/FAST　R- CNN），在检测过程中，只‘看到’候选框内的局部图像信息。因此，若当图像背景（非物体）中的部分数据被包含在候选框中送入检测网络进行检测时，容易被误检测成物体。测试证明，YOLO 对于背景图像的误检率低于 FAST　R- CNN 误检率的一半。

3、通用性强。YOLO 对于艺术类作品中的物体检测同样适用。它对非自然图像物体的检测率远远高于 DPM 和 RCNN 系列检测方法。

但相比 R -CNN 系列物体检测方法，YOLO 具有以下缺点：

1、识别物体位置精准性差。

2、召回率低。

3.1.4 YOLO 算法改进

为提高物体定位精准性和召回率，YOLO 作者提出了 YOLO v2、YOLO9000，这两个版本都是在第一代的基础上做了很多的优化，原来的版本在准确度、速度、容错率上都有所欠缺，而 YOLO_9000 则大大提高训练图像分辨率，引入了 FASTER R-CNN 中 anchor box 的思想，对各网络结构及各层的设计进行了改进，输出层使用卷积层替代 YOLO 的全连接层，联合使用 coco 物体检测标注数据和 imagenet 物体分类标注数据训练物体检测模型。相比 YOLO，YOLO9000 在识别种类、精度、速度、和定位准确性等方面都有大大提升。

3.2 YOLO 算法目标检测实现

3.2.1 输入图像预处理

YOLO 将输入图像分成 S x S 个格子，每个格子负责检测‘落入’该格子的物体。若某个物体的中心位置的坐标落入到某个格子，那么这个格子就负责检测出这个物体；每个格子输出 B 个 bounding box（包含物体的矩形区域）信息，以及 C 个物体属于某种类别的概率信息。

Bounding box 信息包含 5 个数据值，分别是 x,y,w,h,和 confidence。其中 x,y 是指当前格子预测得到的物体的 bounding box 的中心位置的坐标。w,h 是 bounding box 的宽度和高度。注意：实际训练过程中，w 和 h 的值使用图像的宽度和高度进行归一化到[0,1]区间内；x，y 是 bounding box 中心位置相对于当前格子位置的偏移值，并且被归一化到[0,1]。

confidence 反映当前 bounding box 是否包含物体以及物体位置的准确性，计算方式如下：

confidence = P(object)* IOU，其中，若 bounding box 包含物体，则 P(object) = 1；否则 P(object) = 0. IOU(intersection over union)为预测 bounding box 与物体真实区域的交集面积（以像素为单位，用真实区域的像素面积归一化到[0,1]区间），因此，YOLO 网络最终的全连接层的输出维度是 S*S*(B*5 + C)。

3.2.2 核心代码编写

```
# -*- coding:utf-8 -*-
import numpy as np
import tensorflow as tf
import cv2
def leak_relu(x, alpha=0.1):
    return tf.maximum(alpha * x, x)
class Yolo(object):
    def __init__(self, weights_file, verbose=True):
        self.verbose = verbose
```

```
        # detection params
        self.S = 7    # cell size
        self.B = 2    # boxes_per_cell
        self.classes = ["aeroplane", "bicycle", "bird", "boat", "bottle",
                        "bus", "car", "cat", "chair", "cow", "diningtable",
                        "dog", "horse", "motorbike", "person", "pottedplant",
                        "sheep", "sofa", "train","tvmonitor"]
        self.C = len(self.classes) # number of classes
        # offset for box center (top left point of each cell)
        self.x_offset = np.transpose(np.reshape(np.array([np.arange(self.S)]*self.S*self.B),
                                                [self.B, self.S, self.S]), [1, 2, 0])
        self.y_offset = np.transpose(self.x_offset, [1, 0, 2])
        self.threshold = 0.2    # confidence scores threhold
        self.iou_threshold = 0.4
        #    the maximum number of boxes to be selected by non max suppression
        self.max_output_size = 10
        self.sess = tf.Session()
        self._build_net()
        self._build_detector()
        self._load_weights(weights_file)
    def _build_net(self):
        """build the network"""
        if self.verbose:
            print("Start to build the network ...")
        self.images = tf.placeholder(tf.float32, [None, 448, 448, 3])
        net = self._conv_layer(self.images, 1, 64, 7, 2)
        net = self._maxpool_layer(net, 1, 2, 2)
```

```
net = self._conv_layer(net, 2, 192, 3, 1)
net = self._maxpool_layer(net, 2, 2, 2)
net = self._conv_layer(net, 3, 128, 1, 1)
net = self._conv_layer(net, 4, 256, 3, 1)
net = self._conv_layer(net, 5, 256, 1, 1)
net = self._conv_layer(net, 6, 512, 3, 1)
net = self._maxpool_layer(net, 6, 2, 2)
net = self._conv_layer(net, 7, 256, 1, 1)
net = self._conv_layer(net, 8, 512, 3, 1)
net = self._conv_layer(net, 9, 256, 1, 1)
net = self._conv_layer(net, 10, 512, 3, 1)
net = self._conv_layer(net, 11, 256, 1, 1)
net = self._conv_layer(net, 12, 512, 3, 1)
net = self._conv_layer(net, 13, 256, 1, 1)
net = self._conv_layer(net, 14, 512, 3, 1)
net = self._conv_layer(net, 15, 512, 1, 1)
net = self._conv_layer(net, 16, 1024, 3, 1)
net = self._maxpool_layer(net, 16, 2, 2)
net = self._conv_layer(net, 17, 512, 1, 1)
net = self._conv_layer(net, 18, 1024, 3, 1)
net = self._conv_layer(net, 19, 512, 1, 1)
net = self._conv_layer(net, 20, 1024, 3, 1)
net = self._conv_layer(net, 21, 1024, 3, 1)
net = self._conv_layer(net, 22, 1024, 3, 2)
net = self._conv_layer(net, 23, 1024, 3, 1)
net = self._conv_layer(net, 24, 1024, 3, 1)
net = self._flatten(net)
```

```
        net = self._fc_layer(net, 25, 512, activation=leak_relu)
        net = self._fc_layer(net, 26, 4096, activation=leak_relu)
        net = self._fc_layer(net, 27, self.S*self.S*(self.C+5*self.B))
        self.predicts = net
    def _build_detector(self):
        """Interpret the net output and get the predicted boxes"""
        # the width and height of orignal image
        self.width = tf.placeholder(tf.float32, name="img_w")
        self.height = tf.placeholder(tf.float32, name="img_h")
        # get class prob, confidence, boxes from net output
        idx1 = self.S * self.S * self.C
        idx2 = idx1 + self.S * self.S * self.B
        # class prediction
        class_probs = tf.reshape(self.predicts[0, :idx1], [self.S, self.S, self.C])
        # confidence
        confs = tf.reshape(self.predicts[0, idx1:idx2], [self.S, self.S, self.B])
        # boxes -> (x, y, w, h)
        boxes = tf.reshape(self.predicts[0, idx2:], [self.S, self.S, self.B, 4])
        # convert the x, y to the coordinates relative to the top left point of the image
        # the predictions of w, h are the square root
        # multiply the width and height of image
        boxes = tf.stack([(boxes[:, :, :, 0] + tf.constant(self.x_offset, dtype=tf.float32)) / self.S *
self.width,
          (boxes[:, :, :, 1] + tf.constant(self.y_offset, dtype=tf.float32)) / self.S * self.height,
                                tf.square(boxes[:, :, :, 2]) * self.width,
                                tf.square(boxes[:, :, :, 3]) * self.height], axis=3)
        # class-specific confidence scores [S, S, B, C]
```

```
        scores = tf.expand_dims(confs, -1) * tf.expand_dims(class_probs, 2)
        scores = tf.reshape(scores, [-1, self.C])    # [S*S*B, C]
        boxes = tf.reshape(boxes, [-1, 4])    # [S*S*B, 4]
        # find each box class, only select the max score
        box_classes = tf.argmax(scores, axis=1)
        box_class_scores = tf.reduce_max(scores, axis=1)
        # filter the boxes by the score threshold
        filter_mask = box_class_scores >= self.threshold
        scores = tf.boolean_mask(box_class_scores, filter_mask)
        boxes = tf.boolean_mask(boxes, filter_mask)
        box_classes = tf.boolean_mask(box_classes, filter_mask)
        # non max suppression (do not distinguish different classes)
        # ref: https://tensorflow.google.cn/api_docs/python/tf/image/non_max_suppression
        # box (x, y, w, h) -> box (x1, y1, x2, y2)
        _boxes = tf.stack([boxes[:, 0] - 0.5 * boxes[:, 2], boxes[:, 1] - 0.5 * boxes[:, 3],boxes[:, 0] +
0.5 * boxes[:, 2], boxes[:, 1] + 0.5 * boxes[:, 3]], axis=1)
        nms_indices = tf.image.non_max_suppression(_boxes, scores, self.max_output_size,
self.iou_threshold)
        self.scores = tf.gather(scores, nms_indices)
        self.boxes = tf.gather(boxes, nms_indices)
        self.box_classes = tf.gather(box_classes, nms_indices)
    def _conv_layer(self, x, id, num_filters, filter_size, stride):
        """Conv layer"""
        in_channels = x.get_shape().as_list()[-1]
        weight = tf.Variable(tf.truncated_normal([filter_size, filter_size, in_channels, num_filters],
stddev=0.1))
        bias = tf.Variable(tf.zeros([num_filters,]))
```

```
            # padding, note: not using padding="SAME"
            pad_size = filter_size // 2
            pad_mat = np.array([[0, 0], [pad_size, pad_size], [pad_size, pad_size], [0, 0]])
            x_pad = tf.pad(x, pad_mat)
            conv = tf.nn.conv2d(x_pad, weight, strides=[1, stride, stride, 1], padding="VALID")
            output = leak_relu(tf.nn.bias_add(conv, bias))
            if self.verbose:
                print("Layer %d: type=Conv, num_filter=%d, filter_size=%d, stride=%d,
output_shape=%s" \% (id, num_filters, filter_size, stride, str(output.get_shape())))
            return output
        def _fc_layer(self, x, id, num_out, activation=None):
            """fully connected layer"""
            num_in = x.get_shape().as_list()[-1]
            weight = tf.Variable(tf.truncated_normal([num_in, num_out], stddev=0.1))
            bias = tf.Variable(tf.zeros([num_out,]))
            output = tf.nn.xw_plus_b(x, weight, bias)
            if activation:
                output = activation(output)
            if self.verbose:
                print("Layer %d: type=Fc, num_out=%d, output_shape=%s" \
                % (id, num_out, str(output.get_shape())))
            return output
        def _maxpool_layer(self, x, id, pool_size, stride):
            output = tf.nn.max_pool(x, [1, pool_size, pool_size, 1],
            strides=[1, stride, stride, 1], padding="SAME")
            if self.verbose:
                print(" Layer %d: type=MaxPool, pool_size=%d, stride=%d, output_shape=%s" \
```

```
                % (id, pool_size, stride, str(output.get_shape())))
        return output
    def _flatten(self, x):
        """flatten the x"""
        tran_x = tf.transpose(x, [0, 3, 1, 2])    # channle first mode
        nums = np.product(x.get_shape().as_list()[1:])
        return tf.reshape(tran_x, [-1, nums])
    def _load_weights(self, weights_file):
        """Load weights from file"""
        if self.verbose:
            print("Start to load weights from file:%s" % (weights_file))
        saver = tf.train.Saver(max_to_keep=1)
        saver.restore(self.sess, weights_file)
    def detect_from_file(self, image_file, imshow=True, deteted_boxes_file="boxes.txt",
                         detected_image_file="detected_image.jpg"):
        """Do detection given a image file"""
        # read image
        image = cv2.imread(image_file)
        img_h, img_w, _ = image.shape
        scores, boxes, box_classes = self._detect_from_image(image)
        predict_boxes = []
        for i in range(len(scores)):
            predict_boxes.append((self.classes[box_classes[i]], boxes[i, 0],
                                  boxes[i, 1], boxes[i, 2], boxes[i, 3], scores[i]))
        self.show_results(image, predict_boxes, imshow, deteted_boxes_file, detected_image_file)
    def _detect_from_image(self, image):
        """Do detection given a cv image"""
```

```
        img_h, img_w, _ = image.shape
        img_resized = cv2.resize(image, (448, 448))
        img_RGB = cv2.cvtColor(img_resized, cv2.COLOR_BGR2RGB)
        img_resized_np = np.asarray(img_RGB)
        _images = np.zeros((1, 448, 448, 3), dtype=np.float32)
        _images[0] = (img_resized_np / 255.0) * 2.0 - 1.0
        scores, boxes, box_classes = self.sess.run([self.scores, self.boxes, self.box_classes],
                            feed_dict={self.images: _images, self.width: img_w, self.height: img_h})
        return scores, boxes, box_classes
    def show_results(self, image, results, imshow=True, deteted_boxes_file=None,
                             detected_image_file=None):
        """Show the detection boxes"""
        img_cp = image.copy()
        if deteted_boxes_file:
            f = open(deteted_boxes_file, "w")
        #    draw boxes
        for i in range(len(results)):
            x = int(results[i][1])
            y = int(results[i][2])
            w = int(results[i][3]) // 2
            h = int(results[i][4]) // 2
            if self.verbose:
                print("class: %s, [x, y, w, h]=[%d, %d, %d, %d], confidence=%f" % (results[i][0],
                                    x, y, w, h, results[i][-1]))
                cv2.rectangle(img_cp, (x - w, y - h), (x + w, y + h), (0, 255, 0), 2)
                cv2.rectangle(img_cp, (x - w, y - h - 20), (x + w, y - h), (125, 125, 125), -1)
                cv2.putText(img_cp, results[i][0] + ' : %.2f' % results[i][5], (x - w + 5, y - h - 7),
```

```
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1)
            if deteted_boxes_file:
                f.write(results[i][0] + ',' + str(x) + ',' + str(y) + ',' +
                        str(w) + ',' + str(h)+',' + str(results[i][5]) + '\n')
        if imshow:
            cv2.imshow('YOLO_small detection', img_cp)
            cv2.waitKey(0)
        if detected_image_file:
            cv2.imwrite(detected_image_file, img_cp)
        if deteted_boxes_file:
            f.close()
if __name__ == "__main__":
    yolo_net = Yolo("D:/test/YOLO_small.ckpt")
    yolo_net.detect_from_file("D:/test/pig02.jpg")
```

核心代码主要由一下函数组成：图像预处理函数、建立神经网络函数、预测函数、卷积神经网络函数、全连接层函数、池化层函数、faltten 层函数。

3.2.3 目标检测结果

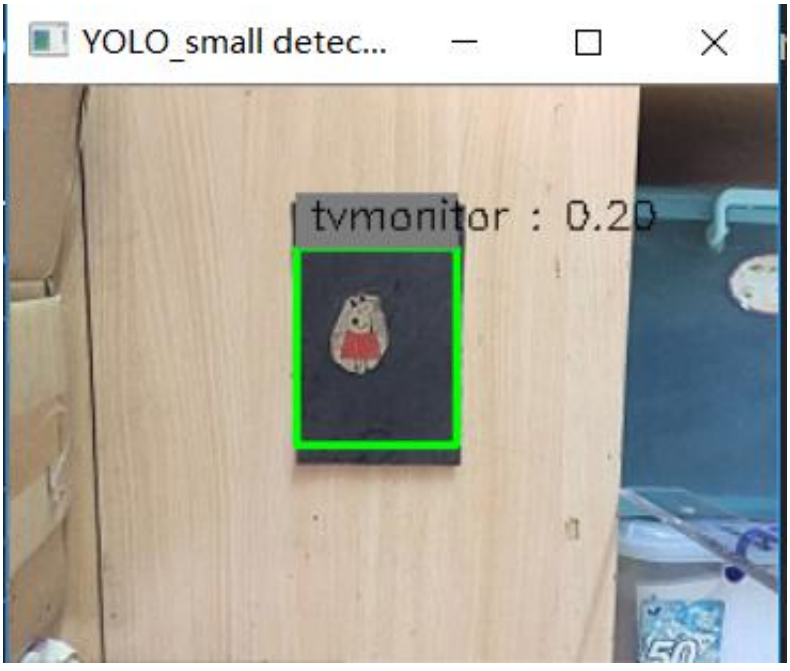

图 3.4 目标检测结果截图

3.3 本章小结

本章重点介绍用于目标检测 YOLO 算法相关原理、优缺点、核心代码以及算法实现结果，从点到线、从线到面依次总体地阐述了 YOLO 算法，我们可以从 YOLO 算法和其他目标检测算法的比较中可以看出，YOLO 算法是目标识别准确率比较高的算法，而 YOLO 算法的进化版 YOLO_9000 正是吸取其他算法的优点，弥补自身算法的缺点而产生，YOLO_9000 在准确率方面已经大大领先其他算法，但是由于本人才识学浅，所以出现不足之处，请老师谅解。

4 基于相似三角形原理的目标测距

4.1 相似三角形目标测距的原理

我们将使用相似三角形来计算相机到一个已知的物体或者目标的距离。

相似三角形就是这么一回事：假设我们有一个宽度为 W 的目标或者物体。然后我们将这个目标放在距离我们的相机为 D 的位置。我们用相机对物体进行拍照并且测量物体的像素宽度 P 。这样我们就得出了相机焦距的公式：

F = (P x D) / W

举个例子，假设我在离相机距离 D = 24 英寸的地方放一张标准的 8.5 x 11 英寸的 A4 纸（横着放；W = 11）并且拍下一张照片。我测量出照片中 A4 纸的像素宽度为 P = 249 像素。

因此我的焦距 F 是：

F = (248px x 24in) / 11in = 543.45

当我继续将我的相机移动靠近或者离远物体或者目标时，我可以用相似三角形来计算出物体离相机的距离：

D’ = (W x F) / P

为了更具体，我们再举个例子，假设我将相机移到距离目标 3 英尺（或者说 36 英寸）的地方并且拍下上述的 A4 纸。通过自动的图形处理我可以获得图片中 A4 纸的像素距离为 170 像素。将这个代入公式得：

D’ = (11in x 543.45) / 170 = 35 英寸

4.2 目标测距实现

4.2.1 核心代码编写

```
# -*- coding:utf-8 -*-
import numpy as np
import cv2

def find_marker(image):
# convert the image to grayscale, blur it, and detect edges
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
        gray = cv2.GaussianBlur(gray, (5, 5), 0)
        edged = cv2.Canny(gray, 35, 125)
        (_,cnts,_) = cv2.findContours(edged.copy(), cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
        c = max(cnts, key = cv2.contourArea)
        # compute the bounding box of the of the paper region and return it
        return cv2.minAreaRect(c)
    def distance_to_camera(knownWidth, focalLength, perWidth):
        # compute and return the distance from the maker to the camera
        return (knownWidth * focalLength) / perWidth
    # initialize the known distance from the camera to the object, which
    # in this case is 24 inches
    KNOWN_DISTANCE = 18.0
    # initialize the known object width, which in this case, the piece of
    # paper is 11 inches wide
    KNOWN_WIDTH = 18.0
    # initialize the list of images that we'll be using
    IMAGE_PATHS = ["D:/test/pp01.jpg","D:/test/pp02.jpg","D:/test/pp03.jpg"]
    # load the furst image that contains an object that is KNOWN TO BE 2 feet
    # from our camera, then find the paper marker in the image, and initialize
    # the focal length
    image = cv2.imread(IMAGE_PATHS[0])
    marker = find_marker(image)
    focalLength = (marker[1][0] * KNOWN_DISTANCE) / KNOWN_WIDTH
    # loop over the images
    if __name__ == "__main__":
        for imagePath in IMAGE_PATHS:
            # load the image, find the marker in the image, then compute the
            # distance to the marker from the camera
            image = cv2.imread(imagePath)
            marker = find_marker(image)
            inches = distance_to_camera(KNOWN_WIDTH, focalLength, marker[1][0])
            # draw a bounding box around the image and display it
            box = np.int0(cv2.boxPoints(marker))
            cv2.drawContours(image, [box], -1, (0, 255, 0), 2)
            cv2.putText(image, "%.3fft" % (inches / 12),
```

```
                (image.shape[1] - 200, image.shape[0] - 20),
cv2.FONT_HERSHEY_SIMPLEX,
                2.0, (0, 255, 0), 3)
        cv2.imshow("image", image)
        cv2.waitKey(0)
```

核心代码中，主要由两个函数组成，分别为 find_marker、distance_to_camera，其中 find_marker 函数是寻找出检测物体的边缘，并进行锁定，而 distance_to_camera 则是计算检测物体到摄像头的距离，距离公式：focal_Length = (marker[1][0] * KNOWN_DISTANCE) / KNOWN_WIDTH

4.2.2 目标测距结果

图 4.1 用于标定系统初始化的 focal_Length(实际距离：25cm)

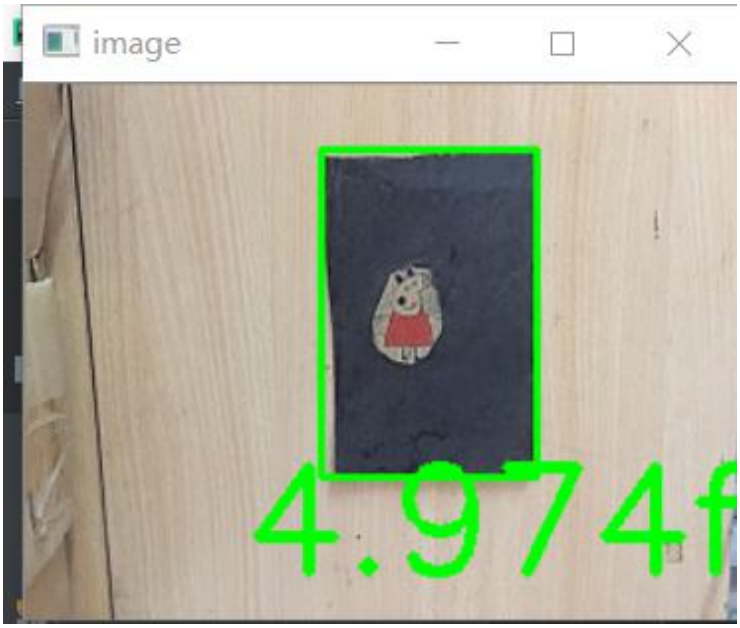

图 4.2 实验结果 1（实际距离：50.00cm，预测距离：49.74cm）

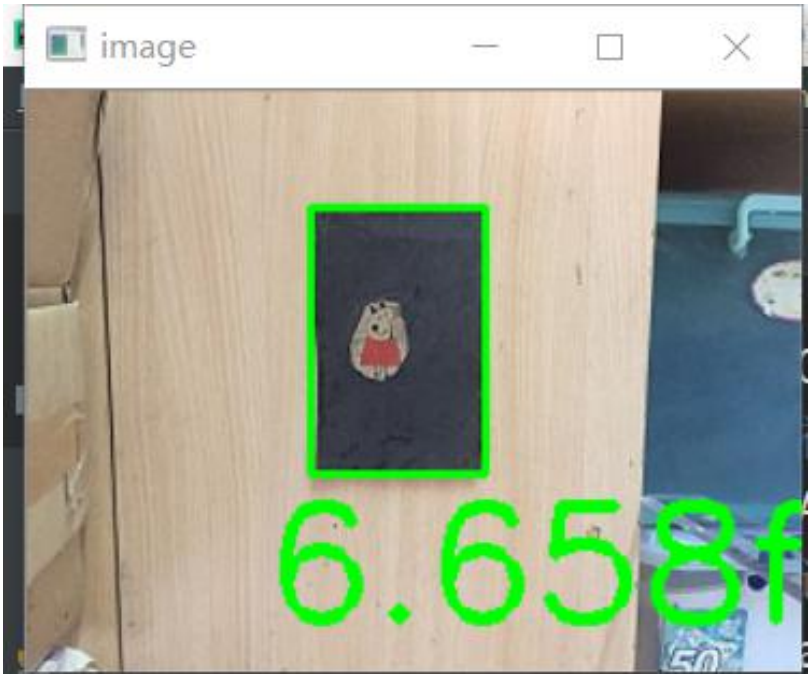

图 4.3 实验结果 2（实际距离：65.00cm，预测距离：66.58cm）

4.3 本章小结

本章主要介绍了利用相似三角形实现目标测距的原理，属于单目视觉方面的内容，并对实验中的核心代码进行相应的介绍，由于本人才识学浅，不足之处，请老师予以指导。

5 结论

本文的目标检测所用到的算法是 YOLO 算法,而目标测距是基于传统的相似三角形目标测距方法，在老师和身边大神同学的帮忙下，和自己不断努力学习下，基本按照老师预定的要求完成毕业课设任务，但是代码的鲁棒性还存在不足之处，在目标检测模块可以考虑升级 YOLO 算法为 YOLO_9000 算法，可以实现更加的准确度和识别度；而在目标测距模块，由于算法的原因，只能考虑弱化环境因素，减少环境对测距结果的干扰，并且提高算法代码的鲁棒性。

参 考 文 献

[1] 王震. 基于深度学习的快速目标检测技术研究[D]. 天津理工大学, 2017.

[2] 韩延祥, 张志胜, 戴敏. 用于目标测距的单目视觉测量方法[J]. 光学精密工程, 2011, 19(05):1110-1117.

[3] 周星, 高志军. 立体视觉技术的应用与发展[J]. 工程图报, 2010, 31(04):50-55.

[4] 温捷文, 战荫伟, 凌伟林, 郭灿樟. 实时目标检测算法 YOLO 的批再规范化处理[J/OL]. 计算机应用研究, 2018(11):1-2[2018-04-25].

[5] 张曙, 华云松. 基于双目立体视觉的目标物定位研究[J]. 软件导刊, 2018, 17(02):198-201.

[6] 吴恩启, 田士强, 徐世鹏, 蒋猛. 单相机立体视觉测量技术的研究[J]. 光学技术, 2018, 44(01):1-5.

[7] 王志, 陈平, 潘晋孝. 基于深度学习的复杂背景下目标检测[J]. 重庆理工大学学报(自然科学), 2018, 32(04):171-176.

[8] 张秀丽, 卓辉, 庞存锁. 高速目标检测性能影响因素的分析方法[J]. 中北大学学报(自然科学版), 2018(02):214-219.

[9] Hongyu Wang,Baomin Zhang,Xun Zhao,Cong Li,Cunyue Lu. A study on low-cost, high-accuracy, and real-time stereo vision algorithms for UAV power line inspection[P]. International Conference on Machine Vision,2018.

[10] Brad Walker.Comparison of the Birds oint-New Madrid Floodway, Mississippi River and the Yolo Bypass, Sacramento River[J].Journal of Earth Science,2016,27(01):47-54.

致　谢

时间飞逝，转眼间四年的大学本科生活即将结束，我要向所有关心、支持、帮助过我的人表示我最真挚的谢意！

首先，我要深深地感谢我的毕业课设指导老师徐胜老师，他为人谦和，平易近人。在论文的选题、搜集资料和写作阶段，他都倾注了极大的关怀和鼓励；本设计论文是在导师徐胜老师的悉心指导下完成的，与此同时，也要感谢帮助我的朋友们，没有你们的倾情指导，我不可能完成这份毕业课设；本论文从选题到完成，每一步都是在导师的指导下完成的，倾注了导师大量的心血。在此谨向导师徐胜老师表示崇高的敬意和由衷的感谢！本设计论文的顺利完成，离不开各位老师、同学和朋友的关心和帮助。在此感谢帮助过我的老师和同学的指导和帮助！另外也感谢我家人，他们的理解和支持使我能够专心完成我的学业；祝愿实验室的老师们，在科研路上硕果累累，阖家欢乐，工作顺利，身体健健康康；身边继续升学的同学们，保持冲劲，再接再厉，为自己心中宏伟的目标而不懈努力，然后再科研之余，也要保重身体，加油！

感谢可爱的自己在大学生活中变得更加勇敢和坚强，并开始能够独立面对挫折和挑战。

最后还要向各位审阅、评议和参与本人论文答辩的所有老师们致以最深切的敬意和感谢！

附录 1 YOLO 算法程序

```
# -*- coding:utf-8 -*-

import numpy as np

import tensorflow as tf

import cv2

def leak_relu(x, alpha=0.1):

    return tf.maximum(alpha * x, x)

class Yolo(object):

    def __init__(self, weights_file, verbose=True):

        self.verbose = verbose

        # detection params

        self.S = 7    # cell size

        self.B = 2    # boxes_per_cell

        self.classes = ["aeroplane", "bicycle", "bird", "boat", "bottle",

                        "bus", "car", "cat", "chair", "cow", "diningtable",

                        "dog", "horse", "motorbike", "person", "pottedplant",

                        "sheep", "sofa", "train","tvmonitor"]

        self.C = len(self.classes) # number of classes

        # offset for box center (top left point of each cell)

        self.x_offset = np.transpose(np.reshape(np.array([np.arange(self.S)]*self.S*self.B),

                                                [self.B, self.S, self.S]), [1, 2, 0])

        self.y_offset = np.transpose(self.x_offset, [1, 0, 2])

        self.threshold = 0.2    # confidence scores threhold

        self.iou_threshold = 0.4

        #    the maximum number of boxes to be selected by non max suppression

        self.max_output_size = 10

        self.sess = tf.Session()
```

```
        self._build_net()
        self._build_detector()
        self._load_weights(weights_file)
    def _build_net(self):
        """build the network"""
        if self.verbose:
            print("Start to build the network ...")
        self.images = tf.placeholder(tf.float32, [None, 448, 448, 3])
        net = self._conv_layer(self.images, 1, 64, 7, 2)
        net = self._maxpool_layer(net, 1, 2, 2)
        net = self._conv_layer(net, 2, 192, 3, 1)
        net = self._maxpool_layer(net, 2, 2, 2)
        net = self._conv_layer(net, 3, 128, 1, 1)
        net = self._conv_layer(net, 4, 256, 3, 1)
        net = self._conv_layer(net, 5, 256, 1, 1)
        net = self._conv_layer(net, 6, 512, 3, 1)
        net = self._maxpool_layer(net, 6, 2, 2)
        net = self._conv_layer(net, 7, 256, 1, 1)
        net = self._conv_layer(net, 8, 512, 3, 1)
        net = self._conv_layer(net, 9, 256, 1, 1)
        net = self._conv_layer(net, 10, 512, 3, 1)
        net = self._conv_layer(net, 11, 256, 1, 1)
        net = self._conv_layer(net, 12, 512, 3, 1)
        net = self._conv_layer(net, 13, 256, 1, 1)
        net = self._conv_layer(net, 14, 512, 3, 1)
        net = self._conv_layer(net, 15, 512, 1, 1)
        net = self._conv_layer(net, 16, 1024, 3, 1)
```

```
        net = self._maxpool_layer(net, 16, 2, 2)
        net = self._conv_layer(net, 17, 512, 1, 1)
        net = self._conv_layer(net, 18, 1024, 3, 1)
        net = self._conv_layer(net, 19, 512, 1, 1)
        net = self._conv_layer(net, 20, 1024, 3, 1)
        net = self._conv_layer(net, 21, 1024, 3, 1)
        net = self._conv_layer(net, 22, 1024, 3, 2)
        net = self._conv_layer(net, 23, 1024, 3, 1)
        net = self._conv_layer(net, 24, 1024, 3, 1)
        net = self._flatten(net)
        net = self._fc_layer(net, 25, 512, activation=leak_relu)
        net = self._fc_layer(net, 26, 4096, activation=leak_relu)
        net = self._fc_layer(net, 27, self.S*self.S*(self.C+5*self.B))
        self.predicts = net
    def _build_detector(self):
        """Interpret the net output and get the predicted boxes"""
        # the width and height of orignal image
        self.width = tf.placeholder(tf.float32, name="img_w")
        self.height = tf.placeholder(tf.float32, name="img_h")
        # get class prob, confidence, boxes from net output
        idx1 = self.S * self.S * self.C
        idx2 = idx1 + self.S * self.S * self.B
        # class prediction
        class_probs = tf.reshape(self.predicts[0, :idx1], [self.S, self.S, self.C])
        # confidence
        confs = tf.reshape(self.predicts[0, idx1:idx2], [self.S, self.S, self.B])
        # boxes -> (x, y, w, h)
```

```
        boxes = tf.reshape(self.predicts[0, idx2:], [self.S, self.S, self.B, 4])
        # convert the x, y to the coordinates relative to the top left point of the image
        # the predictions of w, h are the square root
        # multiply the width and height of image
        boxes = tf.stack([(boxes[:, :, :, 0] + tf.constant(self.x_offset, dtype=tf.float32)) / self.S *
self.width,
                          (boxes[:, :, :, 1] + tf.constant(self.y_offset, dtype=tf.float32)) / self.S *
self.height,
                          tf.square(boxes[:, :, :, 2]) * self.width,
                          tf.square(boxes[:, :, :, 3]) * self.height], axis=3)
        # class-specific confidence scores [S, S, B, C]
        scores = tf.expand_dims(confs, -1) * tf.expand_dims(class_probs, 2)
        scores = tf.reshape(scores, [-1, self.C])    # [S*S*B, C]
        boxes = tf.reshape(boxes, [-1, 4])    # [S*S*B, 4]
        # find each box class, only select the max score
        box_classes = tf.argmax(scores, axis=1)
        box_class_scores = tf.reduce_max(scores, axis=1)
        # filter the boxes by the score threshold
        filter_mask = box_class_scores >= self.threshold
        scores = tf.boolean_mask(box_class_scores, filter_mask)
        boxes = tf.boolean_mask(boxes, filter_mask)
        box_classes = tf.boolean_mask(box_classes, filter_mask)
        # non max suppression (do not distinguish different classes)
        # ref: https://tensorflow.google.cn/api_docs/python/tf/image/non_max_suppression
        # box (x, y, w, h) -> box (x1, y1, x2, y2)
        _boxes = tf.stack([boxes[:, 0] - 0.5 * boxes[:, 2], boxes[:, 1] - 0.5 * boxes[:, 3],
                           boxes[:, 0] + 0.5 * boxes[:, 2], boxes[:, 1] + 0.5 * boxes[:, 3]],
```

```
axis=1)
            nms_indices = tf.image.non_max_suppression(_boxes, scores,
                                                       self.max_output_size,
self.iou_threshold)
            self.scores = tf.gather(scores, nms_indices)
            self.boxes = tf.gather(boxes, nms_indices)
            self.box_classes = tf.gather(box_classes, nms_indices)
        def _conv_layer(self, x, id, num_filters, filter_size, stride):
            """Conv layer"""
            in_channels = x.get_shape().as_list()[-1]
            weight = tf.Variable(tf.truncated_normal([filter_size, filter_size,
                                                      in_channels, num_filters], stddev=0.1))
            bias = tf.Variable(tf.zeros([num_filters,]))
            # padding, note: not using padding="SAME"
            pad_size = filter_size // 2
            pad_mat = np.array([[0, 0], [pad_size, pad_size], [pad_size, pad_size], [0, 0]])
            x_pad = tf.pad(x, pad_mat)
            conv = tf.nn.conv2d(x_pad, weight, strides=[1, stride, stride, 1], padding="VALID")
            output = leak_relu(tf.nn.bias_add(conv, bias))
            if self.verbose:
                print("        Layer %d: type=Conv, num_filter=%d, filter_size=%d, stride=%d,
output_shape=%s" \
                      % (id, num_filters, filter_size, stride, str(output.get_shape())))
            return output
        def _fc_layer(self, x, id, num_out, activation=None):
            """fully connected layer"""
            num_in = x.get_shape().as_list()[-1]
```

```
        weight = tf.Variable(tf.truncated_normal([num_in, num_out], stddev=0.1))
        bias = tf.Variable(tf.zeros([num_out,]))
        output = tf.nn.xw_plus_b(x, weight, bias)
        if activation:
            output = activation(output)
        if self.verbose:
            print("        Layer %d: type=Fc, num_out=%d, output_shape=%s" \
                  % (id, num_out, str(output.get_shape())))
        return output
    def _maxpool_layer(self, x, id, pool_size, stride):
        output = tf.nn.max_pool(x, [1, pool_size, pool_size, 1],
                                strides=[1, stride, stride, 1], padding="SAME")
        if self.verbose:
            print("        Layer %d: type=MaxPool, pool_size=%d, stride=%d, output_shape=%s" \
                  % (id, pool_size, stride, str(output.get_shape())))
        return output
    def _flatten(self, x):
        """flatten the x"""
        tran_x = tf.transpose(x, [0, 3, 1, 2])    # channle first mode
        nums = np.product(x.get_shape().as_list()[1:])
        return tf.reshape(tran_x, [-1, nums])
    def _load_weights(self, weights_file):
        """Load weights from file"""
        if self.verbose:
            print("Start to load weights from file:%s" % (weights_file))
        saver = tf.train.Saver(max_to_keep=1)
        saver.restore(self.sess, weights_file)
```

```
def detect_from_file(self, image_file, imshow=True, deteted_boxes_file="boxes.txt",
                     detected_image_file="detected_image.jpg"):
    """Do detection given a image file"""
    # read image
    image = cv2.imread(image_file)
    img_h, img_w, _ = image.shape
    scores, boxes, box_classes = self._detect_from_image(image)
    predict_boxes = []
    for i in range(len(scores)):
        predict_boxes.append((self.classes[box_classes[i]], boxes[i, 0],
                              boxes[i, 1], boxes[i, 2], boxes[i, 3], scores[i]))
    self.show_results(image, predict_boxes, imshow, deteted_boxes_file, detected_image_file)
def _detect_from_image(self, image):
    """Do detection given a cv image"""
    img_h, img_w, _ = image.shape
    img_resized = cv2.resize(image, (448, 448))
    img_RGB = cv2.cvtColor(img_resized, cv2.COLOR_BGR2RGB)
    img_resized_np = np.asarray(img_RGB)
    _images = np.zeros((1, 448, 448, 3), dtype=np.float32)
    _images[0] = (img_resized_np / 255.0) * 2.0 - 1.0
    scores, boxes, box_classes = self.sess.run([self.scores, self.boxes, self.box_classes],
                feed_dict={self.images: _images, self.width: img_w, self.height: img_h})
    return scores, boxes, box_classes
def show_results(self, image, results, imshow=True, deteted_boxes_file=None,
                 detected_image_file=None):
    """Show the detection boxes"""
    img_cp = image.copy()
```

```
        if deteted_boxes_file:
            f = open(deteted_boxes_file, "w")
        #    draw boxes
        for i in range(len(results)):
            x = int(results[i][1])
            y = int(results[i][2])
            w = int(results[i][3]) // 2
            h = int(results[i][4]) // 2
            if self.verbose:
                print("      class: %s, [x, y, w, h]=[%d, %d, %d, %d], confidence=%f" %
(results[i][0],
                                    x, y, w, h, results[i][-1]))
                cv2.rectangle(img_cp, (x - w, y - h), (x + w, y + h), (0, 255, 0), 2)
                cv2.rectangle(img_cp, (x - w, y - h - 20), (x + w, y - h), (125, 125, 125), -1)
                cv2.putText(img_cp, results[i][0] + ' : %.2f' % results[i][5], (x - w + 5, y - h - 7),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1)
            if deteted_boxes_file:
                f.write(results[i][0] + ',' + str(x) + ',' + str(y) + ',' +
                        str(w) + ',' + str(h)+',' + str(results[i][5]) + '\n')
        if imshow:
            cv2.imshow('YOLO_small detection', img_cp)
            cv2.waitKey(0)
        if detected_image_file:
            cv2.imwrite(detected_image_file, img_cp)
        if deteted_boxes_file:
            f.close()
if __name__ == "__main__":
```

```
    yolo_net = Yolo("D:/test/YOLO_small.ckpt")
    yolo_net.detect_from_file("D:/test/pig02.jpg")
```

附录 2 目标测距算法程序

```
# -*- coding:utf-8 -*-
import numpy as np
import cv2


def find_marker(image):
# convert the image to grayscale, blur it, and detect edges
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, (5, 5), 0)
    edged = cv2.Canny(gray, 35, 125)
    (_,cnts,_) = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
    c = max(cnts, key = cv2.contourArea)
    # compute the bounding box of the of the paper region and return it
    return cv2.minAreaRect(c)
def distance_to_camera(knownWidth, focalLength, perWidth):
    # compute and return the distance from the maker to the camera
    return (knownWidth * focalLength) / perWidth
# initialize the known distance from the camera to the object, which
# in this case is 24 inches
KNOWN_DISTANCE = 18.0
# initialize the known object width, which in this case, the piece of
# paper is 11 inches wide
```

```
KNOWN_WIDTH = 18.0
# initialize the list of images that we'll be using
IMAGE_PATHS = ["D:/test/pp01.jpg","D:/test/pp02.jpg","D:/test/pp03.jpg"]
# load the furst image that contains an object that is KNOWN TO BE 2 feet
# from our camera, then find the paper marker in the image, and initialize
# the focal length
image = cv2.imread(IMAGE_PATHS[0])
marker = find_marker(image)
focalLength = (marker[1][0] * KNOWN_DISTANCE) / KNOWN_WIDTH
# loop over the images
if __name__ == "__main__":
    for imagePath in IMAGE_PATHS:
        # load the image, find the marker in the image, then compute the
        # distance to the marker from the camera
        image = cv2.imread(imagePath)
        marker = find_marker(image)
        inches = distance_to_camera(KNOWN_WIDTH, focalLength, marker[1][0])
        # draw a bounding box around the image and display it
        box = np.int0(cv2.boxPoints(marker))
        cv2.drawContours(image, [box], -1, (0, 255, 0), 2)
        cv2.putText(image, "%.3fft" % (inches / 12),
                    (image.shape[1] - 200, image.shape[0] - 20),
cv2.FONT_HERSHEY_SIMPLEX,
                    2.0, (0, 255, 0), 3)
        cv2.imshow("image", image)
        cv2.waitKey(0)
```