
摘要

摘要

das

关键词：无人驾驶，激光雷达，障碍物检测，深度学习，多传感器融合

ABSTRACT

ABSTRACT

With the widespread engineering applications ranging from broadband signals and non-linear systems, time-domain integral equations (TDIE) methods for analyzing transient electromagnetic scattering problems are becoming widely used nowadays. TDIE-based marching-on-in-time (MOT) scheme and its fast algorithm are researched in this dissertation, including the numerical techniques of MOT scheme, late-time stability of MOT scheme, and two-level PWTD-enhanced MOT scheme. The contents are divided into four parts shown as follows.

.....

Keywords: time-domain electromagnetic scattering, time-domain integral equation (TDIE), marching-on in-time (MOT) scheme, late-time instability, plane wave time-domain (PWTD) algorithm

目 录

第1章 绪论	1
1.1 研究工作的背景与意义	1
1.2 无人车三维障碍物检测的国内外研究历史与现状	2
1.3 本文的主要贡献与创新	2
1.4 本论文的结构安排	2
第2章 三维感知传感器机构设计	3
2.1 三维感知传感器机构的机械结构设计	3
2.1.1 机械结构运动原理	4
2.1.2 曲柄连杆机构的设计	5
2.2 三维感知传感器机构的电路设计	7
2.2.1 驱动器	7
2.2.2 执行机构	8
2.2.3 传感器	8
2.2.3.1 角度传感器	8
2.2.3.2 激光雷达	8
2.2.4 主控板	8
2.2.5 电路拓扑	9
2.3 三维感知传感器机构的软件设计与运动控制	9
2.4 本章小结	11
第3章 点云的多帧融合与激光雷达和相机标定	12
3.1 激光雷达点云的多帧融合	12
3.1.1 一种朴素的多帧融合策略	12
3.1.2 点云的运动畸变的形成与矫正	13
3.1.2.1 点云的运动畸变	13
3.1.2.2 运动畸变的矫正	14
3.1.3 纠正运动畸变后的多帧融合策略	16
3.2 激光雷达与相机的标定	17

目录

3.2.1 标定板	17
3.2.2 相机坐标系中的三维特征点提取	18
3.2.3 LiDAR坐标系中的三维特征点提取	19
3.2.4 刚体变换求解	19
3.2.5 多帧刚体变换结合	22
3.3 本章小结	22
第4章 基于视觉与三维点云融合的三维障碍物检测方法	24
4.1 YOLO-一种实时目标检测网络	24
4.2 基于YOLO的视觉、三维点云结合的三维障碍物检测	26
4.2.1 相机成像原理	26
4.2.2 激光雷达点云的投影	28
4.2.3 点云前景与背景的分割	29
4.2.4 目标三维坐标的计算与检测结果的优化	31
4.3 本章小结	32
第5章 实验验证与结果分析	34
5.1 Gazebo下的三维感知机构仿真	34
5.2 三维感知机构结合里程计信息构建三维地图	35
5.3 基于点云投影到深度图像上的物体分割	36
5.3.1 地面点的去除	36
5.3.2 点云到深度图像的投影	38
5.3.3 基于深度图像的物体分割	40
5.4 本章小结	41
第6章 全文总结与后续工作展望	44
6.1 全文总结	44
6.2 后续工作展望	44
参考文献	45
致 谢	45
外文资料原文	46
外文资料译文	48

缩略词表

主要符号表

第1章 绪论

1.1 研究工作的背景与意义

近几年来，自动驾驶技术取得了长足的进步，而其中关键的技术就是多传感器的环境感知与融合。环境感知的一个重要环节便是障碍物检测。目前，虽然基于图像的障碍物检测已经取得了卓有成效的进步，然而相较于三维障碍物检测，二维障碍物检测有以下缺陷：

1. 基于单目相机的障碍物检测没有尺度信息，无法恢复出目标的三维坐标。
2. 基于双目相机的障碍物检测，当基线较短时，测量距离较长（5m以上）的物体时计算出来的距离信息很不准确，而当基线较长时，近处物体的检测又容易出现在两个相机的视野盲区之中，从而导致无法三角化而得出距离信息。

基于上述原因，越来越多的目光聚焦在了基于激光雷达（LiDAR）的三维障碍物检测。LiDAR是Light Detection And Ranging的缩写，中文译作“激光探测与测量”，一般指多线数的三维激光雷达传感器。相较于相机图像，激光雷达的点云拥有以下几点优势：

1. 测量范围广。目前的激光雷达的测量有效距离基本都在0.5-100米左右，远高于双目相机三角测距的适用范围。
2. 测量精度高。激光雷达的测距误差可达厘米级，同样优于双目相机的测距结果。

目前最常见的旋转式激光雷达，其本质是多个激光束旋转后对每个时刻的测距结果进行保留与叠加，最后再以点云的形式发布出去。决定激光雷达的分辨率的一个参数为其激光束的个数，一般称之为激光雷达的线数。目前常用的激光雷达线数有16线、32线、64线等，其中由于低线数的激光雷达生成的点云在测量远距离物体时密度较低。举例来说，当使用16线激光雷达检测到20米处的障碍物时，其16线激光束两两之间的距离可以达到70cm，相当于检测20m处的人时，只能够有两线激光束能够返回距离。因此，低线数激光雷达点云的稀疏性较大地制约了三维障碍物检测任务的准确率。

通常在自动驾驶的无人车系统中会在车的四周装上多个16线的激光雷达进行点云融合，或者直接采用线数更高的激光雷达来做障碍物检测的任务。然而目前三维激光雷达造价不菲，无人车系统中光是64线激光雷达的成本就奖金十万美金，如此高昂的成本在一定程度上限制了无人驾驶汽车的普及与推广。

鉴于上述存在在问题，本文希望能够提出一种基于多帧融合的低线数激光雷达感知机构，使其能够通过增加一个在垂直方向的往复运动，并将该机构上激光雷达的多帧点云融合发布来提高三维点云的稠密性，借而解决低线数激光雷达在障碍物检测问题上由于点云的稀疏性而造成的困难。并且，本文还希望通过融合上述机构发布的点云信息以及相机的图像信息，发挥各个传感器的优势从而提高三维障碍物检测任务的准确率与实现三维障碍物的多类别检测。

1.2 无人车三维障碍物检测的国内外研究历史与现状

根据本文的主要研究方向，下面将对基于激光雷达点云的三维目标分割与检测（3D object segmentation and detection）的研究现状进行调研。

针对三维障碍物的分割问题，目前主要有三类方法来实现

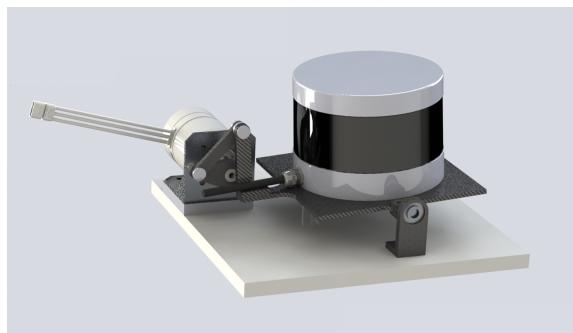
1.3 本文的主要贡献与创新

1.4 本论文的结构安排

第2章 三维感知传感器机构设计

为了能够解决低线数激光雷达在障碍物检测问题上由于点云的稀疏性而造成的困难，本章提出了一种三维感知传感器机构，通过增加三维激光雷达在垂直方向的往复旋转，同时融合多帧激光点云，来增加激光雷达在铅垂方向上的分辨率，实现类似于高线数激光雷达的稠密点云。

2.1 三维感知传感器机构的机械结构设计



(a)



(b)

图 2-1 机构总图(a)solidworks渲染图;(b)实物图

2.1.1 机械结构运动原理

本章所述的机构结构如图2-1(b)所示，其中3508电机提供驱动转矩，曲柄连杆装置将电机的旋转运动转化为激光雷达底座在航向角(yaw)方向上的往复运动，同时绝对值磁编码器记录激光雷达在航向角上的角度变化，以供多传感器融合时使用。

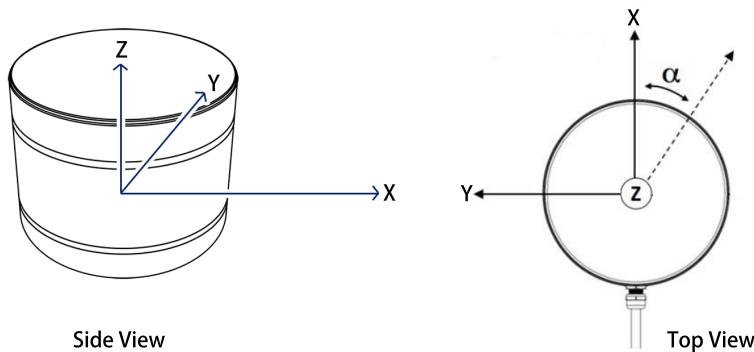


图 2-2 激光雷达坐标系

激光雷达自身的坐标系如图2-2所示，为右手系，其坐标原点在激光雷达的体心。由于激光雷达的传感器性质，导致其在Z轴方向上的点云十分稀疏，因此，本文希望通过机构让激光雷达绕Y轴（也就是上文所述的航向角方向）做往复旋转运动，并通过注册多帧激光雷达的点云来实现激光雷达点云在Y轴方向上的稠密化。

在著名的激光SLAM算法LOAM^[?]中，由于当时的条件限制，其文章作者没有三维激光雷达来进行SLAM，而是用一个舵机给一个二维激光雷达进行竖直方向的往复旋转来增加线数，如图2-3所示。

该机构最大的优点就是结构简单，这也是本文初次采用的机构设计。然而该机构有以下几个缺点：

1. 在往复运动中，当运动方向发生改变时，由于舵机控制精度的问题，很难做到平滑换向，并且经常伴随有较大的震动，给之后的传感器融合算法带来了困难。

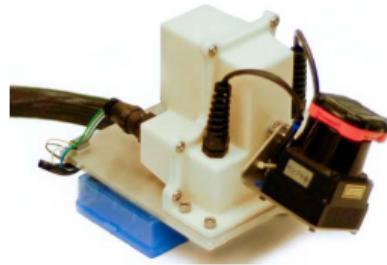


图 2-3 LOAM中的机构设计

2.LOAM中采用的是二维激光雷达，重量较轻，而本文需要带动三维激光雷达进行往复运动，重量较重（近1kg），长时间使用舵机带动会使舵机产生较为明显的回程间隙，影响角度的测量与后续的传感器融合的效果。

因为上述原因，我们没有采用这种结构设计，而是采用了之前提到的曲柄连杆机构来针对三维激光雷达进行往复运动，相较于上述机构，曲柄连杆结构有以下几个优点：

- 1.换向平滑。执行电机只需要一直向同一方向旋转，曲柄连杆机构就能够自动换向，并且输出的角度曲线近似正弦曲线。
- 2.对执行机构负担小。仅需要较小并且较为恒定的转矩就能够驱动较大的负载做往复运动。
- 3.对执行机构的控制要求低。在该机构中，无刷电机只需要输出恒定的转矩就能够完成三维激光雷达在偏航角方向上的往复运动，并且经过验证，其角度输出近似正弦曲线，而若采用上述的舵机机构，要想得到相近的角度曲线，则对舵机的软件控制提出了较高的要求。

综上，本文选择曲柄连杆机构作为该机构的驱动机构。

2.1.2 曲柄连杆机构的设计

该三维感知机构的一个难点在于如何设计与电机相连的曲柄连杆机构。这里参照《机械设计基础》[2]一书中的相应章节对曲柄四连杆机构的连杆长度进行求解。

如图2-4所示，假设已知该铰链四杆机构两连架杆AB和CD所形成的角度 ψ_1 和 ϕ_1 在三个不同位置下的角度，要求连杆a、b、c、d的尺寸。则根据向量向x、y轴投影，有

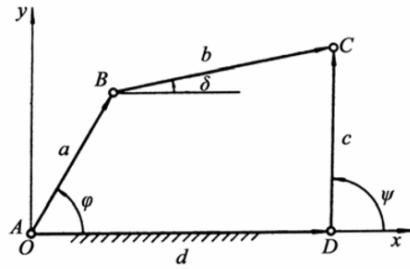


图 2-4 四杆机构的数学模型

$$a \cos \phi + b \cos \delta = d + \cos \phi$$

$$a \sin \phi + b \sin \delta = c \sin \phi$$

将上两式先进行移项，然后作平方和相加，从中消去 δ 后整理有

$$b^2 = a^2 + c^2 + d^2 + 2cd \cos \psi - 2ad \cos \phi - 2ac \cos(\phi - \psi)$$

我们设

$$\begin{cases} R_1 = (a^2 + d^2 + c^2 - b^2) \\ R_2 = d/c \\ R_3 = d/a \end{cases}$$

代入，则上一个式子可以化简为

$$R_1 - R_2 \cos \phi + R_3 \cos \psi = \cos(\phi - \psi)$$

这个式子即为铰链四连杆机构的角位置方程，该方程有三个待定参数 R_1 、 R_2 、 R_3 。故应有三组对应的 ψ 和 ϕ 角才能得出这个方程的解。将三组 ψ_1 和 ϕ_1 角代入求解该方程后，可以得到四个构件之间的长度关系为

$$\begin{cases} a = d/R_3 \\ c = d/R_2 \\ b = \sqrt{a^2 + c^2 + d^2 - 2acR_1} \end{cases}$$

则根据机构的具体设置情况，知道 a, b, c, d 中的任何一条边的长度后，便可知剩下四条边的长度。

在实际设计中，我们已知 ψ_1 和 ϕ_1 的三组对应角度为

$$\begin{cases} \psi_1 = 30^\circ \quad \phi_1 = 36.3^\circ \\ \psi_1 = 60^\circ \quad \phi_1 = 43.87^\circ \\ \psi_1 = 120^\circ \quad \phi_1 = 35.75^\circ \end{cases}$$

并且根据我们的机构设置，构件 d 的长度为105.72mm。将这些已知量代入公式中可得

$$\begin{cases} a = 31.6mm \\ b = 49.18mm \\ c = 108.37mm \end{cases}$$

由此，便得到了曲柄机构的连杆构件设计参数。

2.2 三维感知传感器机构的电路设计

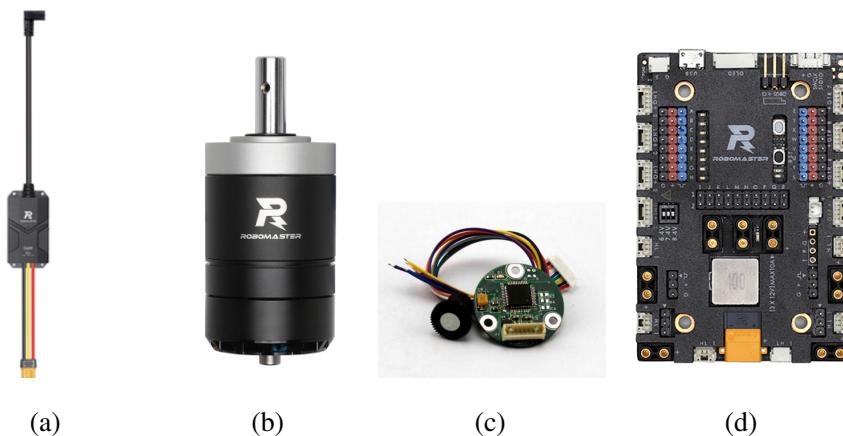


图 2-5 驱动器与传感器(a)C620 电调;(b)M3508无刷电机;(c)磁编码器;(d)RM A型开发板

2.2.1 驱动器

该三维感知机构采用的驱动器为DJI C620电调，如图2-5(a)所示。该电调支

持50-500Hz的PWM（脉宽调制）信号控制以及CAN总线指令控制，最高支持20A的持续电流，支持对CAN总线上的电调快速设置ID，支持通过CAN总线获取电机温度、转子位置和转子速度等信息，切换电机时可无需进行位置传感器的参数校准。

2.2.2 执行机构

该三维感知机构采用的执行机构为DJI M3508无刷电机，如图2-5(b)所示。该电机可搭配上文所述C620电调实现正弦驱动，相比传统方波驱动具有更高的效率、机动性和稳定性。其最高可持续输出力矩为2.8Nm，满足驱动曲柄四连杆机构的需求。

2.2.3 传感器

2.2.3.1 角度传感器

该三维感知机构采用的角度传感器为傲蓝13线磁编码器，如图2-5(c)所示。该编码器采用RS485方式通信，其单圈分辨率为8192cpr，精度为±0.1度。

该编码器为绝对值式编码器，其相对于增量式编码器不同点在于，增量式编码器以上电时的位置为零点，每次使用都要机械对位；而绝对值式编码器能够记录机构的唯一位置，即单圈内编码器的每一个示值，都唯一对应了空间中机构的位置与角度。考虑到我们曲柄连杆机构的特性，显然绝对值式编码器更加符合我们的要求。

2.2.3.2 激光雷达

该三维感知机构采用的激光雷达为速腾聚创的RS-LiDAR-16，如图2-6所示。该激光雷达为16线激光雷达，其测距范围为50cm-150m，精度误差为±2cm。垂直视场角为30度，其角分辨率为2度；水平视场角为360度，其角分辨率为0.09-0.36度（对应的点云频率为5Hz-20Hz）。

2.2.4 主控板

该三维感知机构采用的主控板为DJI Robomaster A型开发板，如图2-5(d)所示。该开发板具备类型丰富的接口，包括12V、5V、3.3V电源接口、CAN接口、UART接口、可变电压PWM接口、SWD接口等。同时该开发板拥有电源输入的防反接、过压保护、缓启动、12V电源输出过流保护、PWM端口的ESD等多重保护。



图 2-6 速腾16线激光雷达

2.2.5 电路拓扑

该三维感知机构的电路拓扑如图2-7所示。激光雷达通过千兆网接口将点云传输到mini PC上，磁编码器通过485转USB与mini PC通信，同时主控通过PWM控制电调输出，调节M3508电机的转速，M3508电机提供曲柄四连杆机构的驱动力矩，而磁编码器又将曲柄机构作用在激光雷达底座上的旋转通过485通信输出到mini PC。

2.3 三维感知传感器机构的软件设计与运动控制

根据上文所述的机械设计以及电路设计，该三维感知机构的软件设计主要实现了以下几个任务：

- 1.实现了各个传感器、主控到Mini PC的通信，同时将数据以ROS（Robot Operating System）话题的方式发布出去，以供第二章节提到的多帧融合算法使用。
- 2.实现了电机的多档调速功能。为了应对不同的场景，在主控中实现了多档调速功能，以调节曲柄机构的往复运动频率。
- 3.实时检测电调的温度信息，提供了基于温度检测的堵转保护（温度过高自动切断控制）。

此外，本文还记录了在电机输出恒定转速情况下的曲柄连杆机构的输出的角度信息，如图2-8所示。该图纵坐标为角度制的输出角度。从图中可以看出，本章

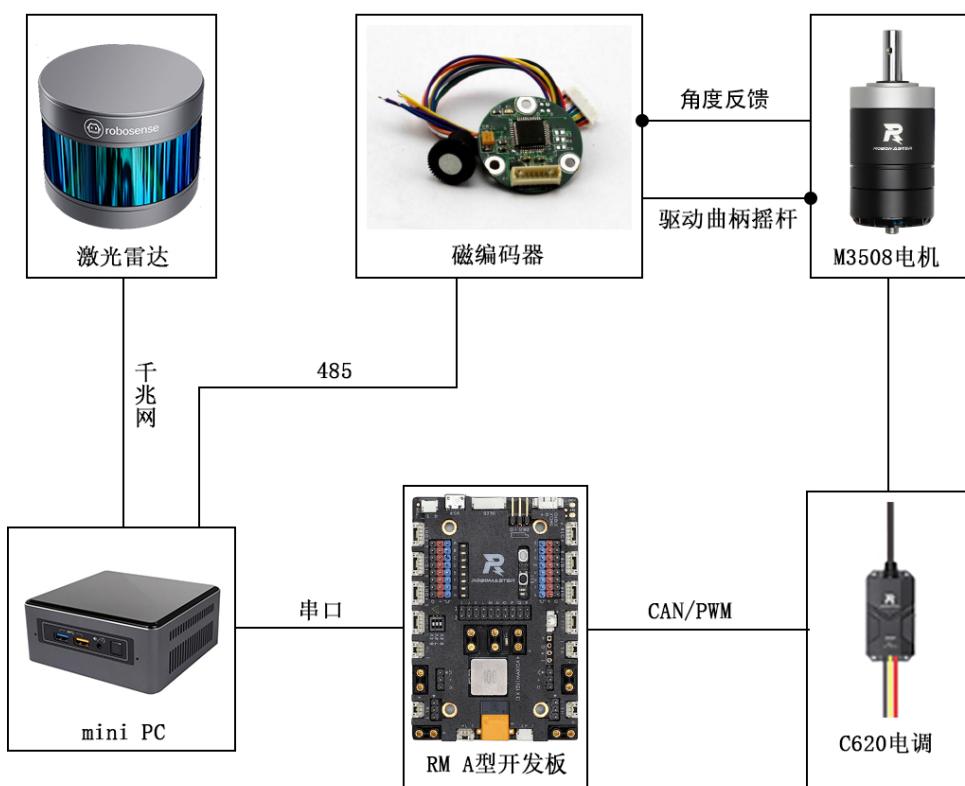


图 2-7 电路拓扑

所设计的三维感知机构其输出角度近似正弦曲线，并且没有较大的换向震动，相比起上文所提到的舵机的结构拥有稳定可靠的优势。

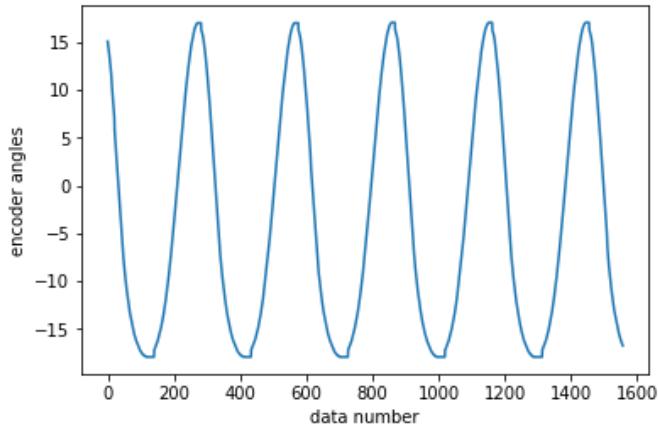


图 2-8 曲柄连杆机构输出角度

2.4 本章小结

本章提出了一种新的三维感知机构，并从该机构的机械设计、电路设计以及软件设计和运动控制三个方面介绍了该机构。在机械方面，本文提出使用无刷电机+曲柄摇杆机构来代替简单的舵机给三维激光雷达提供一个竖直航向角方向上的往复运动，这种结构的优势在于机构换向流畅、控制简单以及对机构负载小，能够为本文后续章节提到的激光雷达的多帧融合提供结构上的稳定与可靠性。在电路方面，本章利用绝对值式磁编码器对曲柄机构运动的角度进行了记录与输出，相较于增量式编码器，磁编码器不需要保证每次上电时机构都在同一个位置，为机械结构的设计提供了便利。在软件方面，本章实现了各个传感器与主控以及mini PC的通信，主控对无刷电机的多档控制以及对电机的堵转保护，并且绘制了输出角度，验证了机构的可行性。本文的后续章节将利用该机构输出的点云信息以及角度信息来进行点云的多帧融合稠密化，并且在融合的点云上进行三维障碍物的检测与分类。

第3章 点云的多帧融合与激光雷达和相机标定

根据本文第二章所提到的机构，能够将三维激光雷达在其yaw角方向上提供一个有规律的正弦往复运动。本文提出该机构的主要目的为将激光雷达在时间轴上的多帧点云进行融合，进而增加激光雷达在竖直方向的分辨率，达到近似于给激光雷达增加线数的效果。本章将对上述机构得到多帧激光雷达点云进行融合，并且在目前机构的基础上进行激光雷达与相机的标定，从而使得融合后的点云能够应用于第四章提到的基于视觉与激光融合的三维障碍物检测方法。

3.1 激光雷达点云的多帧融合

3.1.1 一种朴素的多帧融合策略

点云的注册（registration）是指将有重合部分的点云进行对齐的一项技术。其关键核心为求出给定点云的坐标系相对于目标点云所在坐标系的旋转与平移，借以将给定点云转换到目标点云坐标系中，丰富目标点云的信息，以给之后的点云分割与分类任务提供便利。

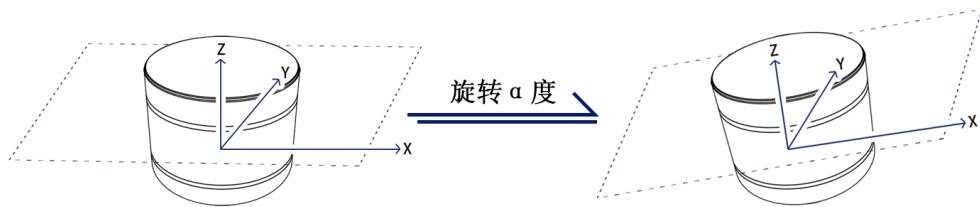


图 3-1 三维感知机构的旋转

如图3-1所示，本文认为当激光雷达航向角与地面平行时，其激光雷达坐标系为参照系 f ，第二章所述的三维感知机构目的就在于在激光雷达做往复运动时，将激光雷达每帧点云在参照系 f 上进行注册，最后在曲柄机构的一个运动周期后将点云融合输出。当机构运动时，激光雷达自身坐标相对于 f 发生了旋转，因而在激光雷达坐标系中的点云相对于 f 有一个 α 角度的旋转量。因而要将此时的点云注册到 f 坐标系中时，要抵消因机构旋转而造成的坐标系的旋转。

最为直观的策略就是，读取绝对值编码器返回的角度 α ，将激光雷达每帧点云沿着航向角方向旋转 $-\alpha$ 的角度，然后注册多帧的激光雷达点云并发布。

在实际的实现过程中，编码器返回角度的频率约为30Hz，而点云发布的频率约为10Hz，在将点云旋转 $-\alpha$ 角度时，对 α 角进行了线性插值以便获得更加精确的结果。同时，根据计算曲柄机构的角度是增加还是减少，来判断曲柄机构的运动方向，并且将曲柄机构运动角度为一个正弦周期内的点云融合为一帧新的点云输出。

然而这种朴素的融合策略在实际中效果不好，体现在融合后的点云所显示的物体轮廓失真严重，如图3-4(a)所示，原因是没有考虑激光雷达的运动对激光雷达点云生成的影响。

3.1.2 点云的运动畸变的形成与矫正

在激光雷达点云的多帧融合中，如果只是进行简单的历史点云叠加（如上文所示），那么融合后的点云相较于真实情况会有很严重的失真，其原因就在于第二章所提到的三维感知机构在给激光雷达在偏航角方向上的往复运动时，点云会产生不可忽视的运动畸变。本章节首先介绍什么是激光雷达点云的运动畸变，然后提出一种通过插值的方式矫正激光雷达的运动畸变。

3.1.2.1 点云的运动畸变

激光雷达的点云的形成本质上是由激光雷达内部的多个激光测距器将一个旋转周期内的各个测量值记录下来并同时发布后得到的。因此点云中的每个点并不是在同一时刻被测量出来的。如果激光雷达在测量的过程中也在运动，那么激光雷达的点云可能会发生畸变^[2]。

下面以二维激光雷达为例，介绍激光雷达点云运动畸变的形成。

图3-2(a)中的黑色的线条表示二维激光雷达处在的真实环境的轮廓图，箭头表示二维激光雷达的运动方向。图3-2(b)中的蓝色的线条表示二维激光雷达的原始

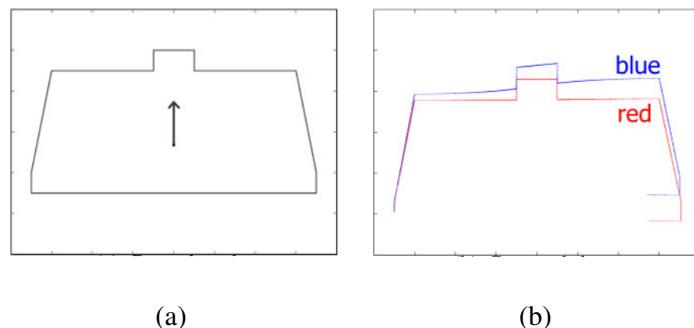


图 3-2 激光雷达运动畸变(a)ground truth;(b)采集得到的数据

数据。注意到其已经发生了畸变，因为二维激光雷达内的激光测距器通过逆时针的方向旋转，使得右上角的数据优先得到，而左上角的数据在激光雷达向箭头方向运动了一段距离之后才进行测量，自然导致了运动畸变的产生。

值得一提的是上图表示的二维激光雷达发生的运动畸变是当激光雷达运动方向为水平运动方向时造成的，而当激光雷达在空间中有垂直方向的旋转运动时，其造成的运动畸变远比水平运动严重。这是因为激光雷达旋转一周的时间普遍在0.1秒左右，其水平运动的距离往往很小可以忽略不计。而当激光雷达有竖直方向的旋转时，即使在0.1s内只有2度的航向角的旋转（这在本文的机构中并不算很快），在测量20m处的物体时，其运动造成的点云畸变可使得点云的同一线上的第一个点与最后一个点的垂直相差将近70cm。

综上所述，对于第二章所述的机构，由于其施加了在激光雷达航向角方向上的旋转，因此导致其在竖直方向上的运动畸变不可忽视，从而简单的叠加点云会导致在做激光雷达的物体检测时的失真。

3.1.2.2 运动畸变的矫正

对于本文所提到的三维感知机构在航向角方向上的旋转所产生的运动畸变的矫正，一个较为朴素的方法是，依次遍历激光雷达每帧点云中的每个点，计算其产生的时间戳 t ，对磁编码器的角度进行插值，计算出在时间戳 t 上的角度 α ，然后将该点绕原点在航向角方向旋转 $-\alpha$ 的角度。这个方法最为直观，然而激光雷达每帧点云高达数十万个点，如果对每个测量得到的点进行插值，则在一秒内要进行近百万次的插值与旋转操作，显然对于无人驾驶汽车上的移动处理器平台来说这是不现实的。

因此本文提出一个假设，设点云中第一个点产生的时间戳为 t_0 ，最后一个点产生的时间戳为 t_k ，则将 t_0 到 t_k 之间的时间均匀分为 n 份，每份长为 Δt 。本文假设 t_0 至 $t_0 + \Delta t$ 、 $t_0 + \Delta t$ 至 $t_0 + 2\Delta t$... $t_k - \Delta t$ 至 t_k 这些时间段，每个时间段内的激光雷达的测距点的产生时间都是相同的，为其第一个点的产生时间。根据这个假设，每个时间段内的所有点都只要进行相同角度变换即可进行运动畸变的矫正。遵循该假设，则每帧点云只需要进行 n 次角度插值即可，极大地减小了运算量。同时虽然该假设认为同一时间段内的点云是同一时间产生的，每个时间段内的点仍有运动畸变的影响，然而在实验过程中发现，只要当 n 取一个不太小的值（ $n \geq 50$ ），则该假设的所产生的时间段内的运动畸变产生的影响很小，可忽略不计。

在实际的程序实现中，由于激光雷达每帧点云是分段传输的，如图3-3所示。RS-LiDAR-16激光雷达采用UDP协议向PC传输点云信息，而UDP协议相较于TCP协

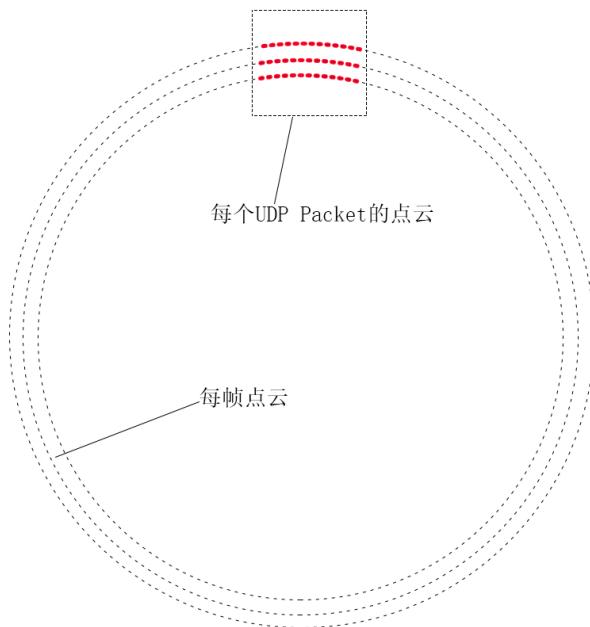


图 3-3 点云的传输

议，发送数据之前不需要双方建立链接，并且发送的数据没有校验，也没有丢包的检测，因此不适合一次性发送大量数据给PC。该激光雷达将一帧点云分为84个UDP包(UDP Packet)，每个包中的点云都是激光雷达旋转 $360/84 = 4.28$ 度后得到的16线的点云的集合。当激光雷达的驱动程序接收到84个UDP Packet之后，将这84个Packet合并成一帧点云输出。

本文修改了RS-LiDAR-16的ROS 驱动程序，将每个UDP Packet不经过合并直

接发布出去，同时在多帧融合的程序里，对每个Packet（而不是每帧）分别进行编码器角度的插值与点云的旋转，最后再将旋转后的84个Packet进行合并发布。

3.1.3 矫正运动畸变后的多帧融合策略

如上文所言，对单帧点云进行多次插值之后的多帧融合算法流程如Algorithm 1所示；。

在Algorithm 1中，FindeClosestAngles函数查找出所有的编码器角度中，时间上离packet的时间戳最近的两帧编码器的角度值，随后对这两个角度进行线性插值得到packet时间戳下的机构的角度。RotatePointCloudByYaw函数将特定的点云绕激光雷达坐标系原点旋转指定的角度。最后将每个矫正后的packet合并到一个新的矫正后的点云中并发布出去即得到了矫正因机构而产生的运动畸变后的点云。

在图3-4(b)中展示了消除运动畸变后的激光雷达测得的轿车的点云图案。相较于图3-4(a)，其点云图案没有出现明显的失真，并且通过机构进行多帧融合后的点云，能够明显的看出轿车的轮廓细节信息，包括车的反光镜、前挡风玻璃等。而矫正畸变前的轿车点云则很难分辨出这些细节，并且由于运动畸变的作用，其体积明显比矫正后的点云更大一些。由此证明了本文提出的多帧融合算法拥有较好的矫正畸变的效果。

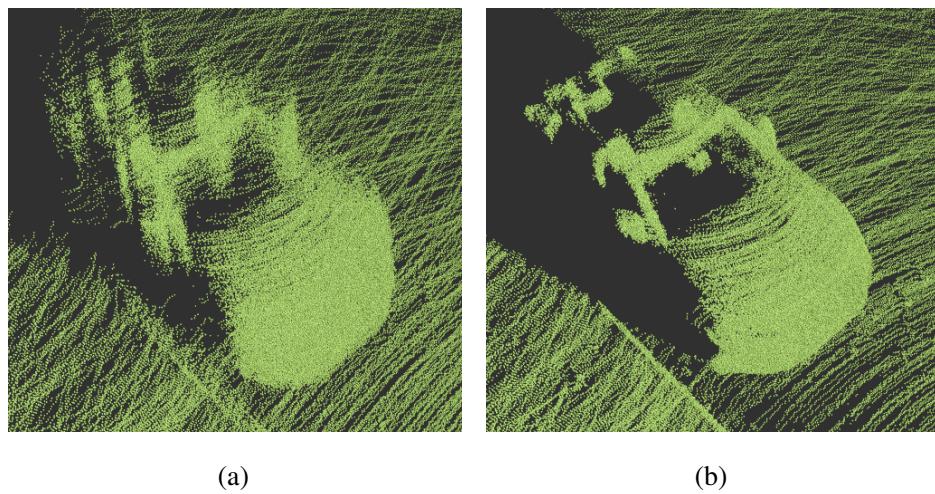


图 3-4 多帧融合后的轿车点云(a)矫正畸变前;(b)矫正畸变后

Algorithm 1 Improved multiple pointcloud fusion algorithm

```

1: for packet  $\in$  point cloud do
2:    $\alpha_1, \alpha_2 = \text{FindClosestAngles}(\text{encoderAngles}, \text{packet.timestamp})$ 
3:    $\alpha = \alpha_1 + (\alpha_2 - \alpha_1) \times \frac{\text{packet.timestamp} - \alpha_1.\text{timestamp}}{\alpha_2.\text{timestamp} - \alpha_1.\text{timestamp}}$ 
4:   rectifiedPacket = RotatePointCloudByYaw(packet,  $-\alpha$ )
5:   rectifiedPointcloud += rectifiedPacket
6: return rectifiedPointcloud

```

3.2 激光雷达与相机的标定

自动驾驶无人机是一个多传感器的系统，多传感器信息的融合可以使整个无人机系统的决策更加智能。激光雷达虽然能够获取较为精确的点云信息，然而点云信息只包含了三维距离信息。而相机可以通过图像获得大量信息诸如颜色、纹理信息等，但是其受光照与天气条件影响严重，并且从单目图像中无法获取三维结构信息。为了同时收集三维信息与物体的颜色与纹理信息，激光雷达与相机经常进行传感器的数据融合，来为多传感器系统提供更稳定的数据支持。为了进行数据的融合，首先得知道相机坐标系与激光雷达坐标系之间的旋转与平移关系，因此，激光雷达与相机的标定就显得尤为重要了。

本章节后续将介绍一种文献^[?]提到的，利用两张贴有ArUco Marker^[?]的标定板所提供的3d-3d特征匹配的方法，来进行相机与激光雷达的标定。

3.2.1 标定板



图 3-5 标定环境

本文参照文献^[?]提到的方法，制作了两块长约40cm，宽约27cm的长方形硬纸板材质的标定板，并且将两块ArUco Marker粘在硬纸板的固定位置上，如图3-5所示。虽然一块标定板已经可以得到四组3d-3d匹配点来解决标定问题，本文仍然采用了两块标定板，目的是构造多于四对的匹配点来减小标定误差。

3.2.2 相机坐标系中的三维特征点提取

ArUco markers是一种经过特定编码的二维码图案，用以实现对二维码自身的定位与畸变矫正。更多细节可以参考文献^[?]。该文献提出，通过特定的机器视觉算法检测到marker的四个角点后，可以对marker上的二维码进行解码运算，进而求得二维码的id与四个角点的顺序。而通过输入marker的边长后，还能够通过PnP^[?]求解出相机坐标系到marker自身坐标系的转换。

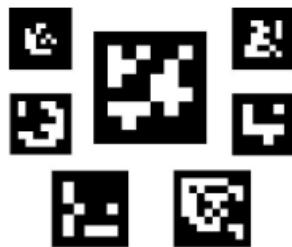


图 3-6 ArUco markers

在本文中，将ArUco marker粘在硬纸板的矩形标定板上，并且测量得出硬纸板的四条边长以及marker在硬纸板中的位置，即可得到硬纸板的四个角点在marker坐标系中的位置。而本文通过ROS中aruco_ros以及aruco_mapping^[?]两个程序包可以检测ArUco marker的位置，进而得到相机坐标系到marker坐标系的转换，从而得到相机坐标系下的硬纸板的四个角点的位置。相机坐标系下角点位置的计算公式为

$$\begin{bmatrix} x_{camera} \\ y_{camera} \\ z_{camera} \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{aruco} \\ y_{aruco} \\ z_{aruco} \\ 1 \end{bmatrix} \quad (3-1)$$

其中， x_{camera} 是相机坐标系中角点的坐标， x_{aruco} 是ArUco marker坐标系中的角点坐标，而上式中的 $[R|t]$ 矩阵则是相机坐标系相对于marker坐标系的欧式变换矩阵。

3.2.3 LiDAR坐标系中的三维特征点提取

本文所参照的标定方法，其在LiDAR坐标系中的三维特征点提取是通过直线拟合的方法进行的。如图3-7所示。图中显示的点为激光雷达的点云投影到相机图像上之后进行边缘检测后所形成的点。在得到该幅图像后，需要手动框选标定板的每条边上的点，随后标定程序会对这些点进行直线拟合，每两条直线的交点即为所求的LiDAR坐标系中的三维点。

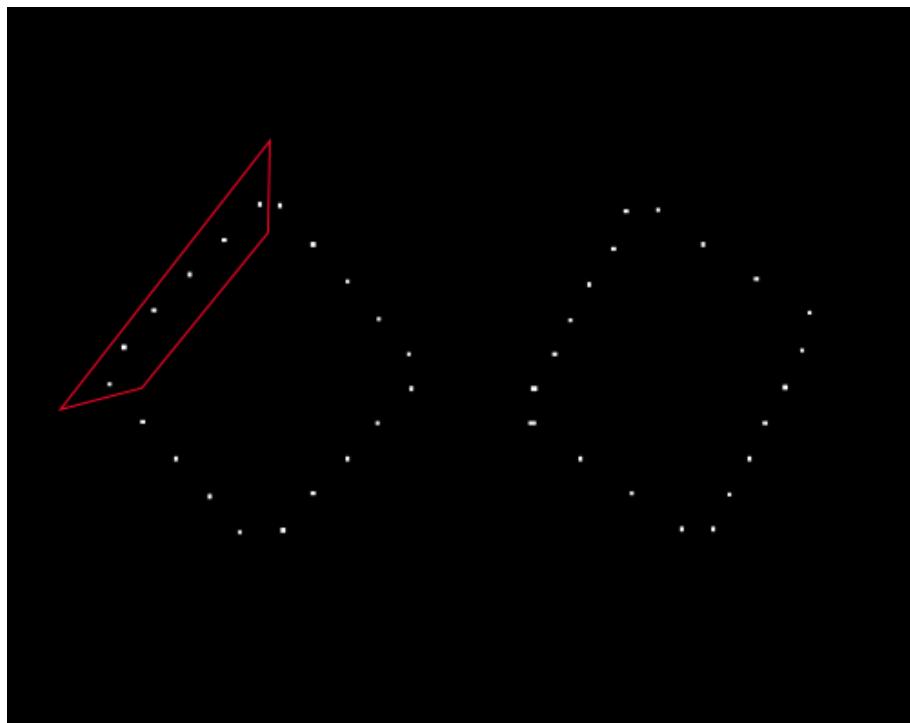


图 3-7 直线拟合提取特征点

3.2.4 刚体变换求解

在得到两个坐标系下的三维特征匹配点之后，两个坐标系之间的 $[R|t]$ 刚体变换矩阵可以通过使用迭代最近点（Iterative Closest Point, ICP）^[?]算法求得。假设 P , Q 为 \mathbb{R}^3 中的一组对应点，ICP算法尝试使经过刚体变换后的 P 点集与 Q 点集的重合误差最小。其求解问题可以表述为式3-2所示。

$$\arg \min_{R \in SO(3), t \in \mathbb{R}^3} \| (RP + t) - Q \|^2 \quad (3-2)$$

一般来说，ICP问题认为对于点集 P 中的每个点，在点集 Q 中与之对应的点为其距离最近的点，依次确认匹配关系后，该算法通过减小两个点集的欧氏距离来对齐两个点集。

ICP的方法找到两组点的匹配关系，在没有良好的初始值的情况下，极易发生误匹配，导致算法最后没办法收敛到一个正确的解。考虑到在本文的标定环境下，两组点的对应关系是已知的，则两组点之间的 $[R|t]$ 即存在一个闭式解。Kabsch算法^{[?][?]}提供了找到两组对应点间的旋转矩阵的闭式解的方法，而两组点间的平移量可以在进行旋转对齐后求得。下面参照文献^[?]，介绍Kabsch算法的主要步骤。

首先假设旋转已知，求两个点集 P 与 Q 之间的平移量，设

$$F(t) = \sum_{i=1}^n \| (RP_i + t) - Q_i \|^2 \quad (3-3)$$

对上式进行求导，令两边等于0，得

$$\frac{\partial F(t)}{\partial t} = 2 \sum_{i=1}^n ((RP_i + t) - Q_i) = 0 \quad (3-4)$$

因为

$$\frac{\partial F(t)}{\partial t} = 2R \sum_{i=1}^n P_i + 2nt - 2 \sum_{i=1}^n Q_i \quad (3-5)$$

可得

$$t = \frac{1}{n} \sum_{i=1}^n Q_i - R \frac{1}{n} \sum_{i=1}^n P_i \quad (3-6)$$

$$t = \overline{Q} - R\overline{P} \quad (3-7)$$

将式3-7的结果代入式3-3中，有

$$R = \arg \min_{R \in SO(3)} \sum_{i=1}^n \| (R(P_i - \overline{P}) - (Q_i - \overline{Q})) \|^2 \quad (3-8)$$

令

$$X = P_i - \overline{P}, X' = RX, Y = Q_i - \overline{Q}$$

则目标函数可化简为

$$\sum_{i=1}^n \| X'_i - Y_i \|^2 = Tr((X' - Y)^T (X' - Y)) \quad (3-9)$$

使用矩阵的迹的性质，上式可化简为

$$Tr((X' - Y)^T(X' - Y)) = Tr(X'^T X') + Tr(Y^T Y) - 2Tr(Y^T X) \quad (3-10)$$

考虑到 R 是一个旋转矩阵，旋转矩阵必定正交且行列式为1，也就是说， $\|X'_i\|^2 = \|X_i\|^2$ ，则有

$$Tr((X' - Y)^T(X' - Y)) = \sum_{i=1}^n (|X_i|^2 + |Y_i|^2) - 2Tr(Y^T X) \quad (3-11)$$

可知 $\sum_{i=1}^n (|X_i|^2 + |Y_i|^2)$ 项与旋转矩阵 R 无关，将其从目标函数中消去，有

$$R = \arg \min_{R \in SO(3)} Tr(Y^T X') \quad (3-12)$$

将 $X' = RX$ 代入，并利用矩阵迹的性质，有

$$Tr(Y^T X') = Tr(Y^T RX) = Tr(XY^T R) \quad (3-13)$$

对 XY^T 进行SVD分解，有 $XY^T = UDV^T$ ，则

$$Tr(XY^T R) = Tr(UDV^T R) = Tr(DV^T RU) = \sum_i^3 d_i v_i^T R u_i \quad (3-14)$$

令 $M = V^T RU$ ，则 M 由正交矩阵相乘而得，故易知其亦为正交矩阵，并且 $\det(M) = \pm 1$ 。因此 M 的每个列向量的模均为1，列向量的每个元素均小于等于1。则有

$$Tr(XY^T R) = \sum_i^3 d_i M_{ii} \leq \sum_i^3 d_i \quad (3-15)$$

令上式值最大，则得到 $M_{ii} = 1$ ，因此 $M = I$ ，为单位矩阵。有

$$M = I \Rightarrow V^T RU = I \Rightarrow R = VU^T \quad (3-16)$$

需要注意的是， R 应当是一个旋转矩阵，也就是说， $R \in SO(3)$ ，所以应当确保 $\det(R) = +1$ 。如果由上式得到的 R ，其特征值为-1，则其为不满足条件的解，需要找到使 $Tr(Y^T X')$ 第二大的 R ，即

$$Tr(Y^T X') = d_1 M_{11} + d_2 M_{22} + d_3 M_{33} \text{ where } d_1 \geq d_2 \geq d_3 \text{ and } |M_{ii}| \leq 1 \quad (3-17)$$

在上式中，当 $M_{11} = M_{22} = 1$ 且 $M_{33} = -1$ 时，该式值第二大。将上述情况考虑进去，则 R 的闭式解为 $R = UCV^T$ ，其中 C 为一个矫正矩阵，

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \text{sign}(\det(UV^T)) \end{bmatrix}$$

3.2.5 多帧刚体变换结合

在实验中发现，由相机图像求得的三维特征点比较稳定，而由于激光雷达本身存在传感器误差，其通过直线拟合求交点得到的三维角点不够稳定，在相机与激光雷达位置均固定的情况下，每次测得的激光雷达测得的三维特征角点在会有一个较小的位移，而这样的位移会对用Kabsch算法得到的解产生较大的误差。

为了减小因激光雷达传感器误差而造成的三维特征角点不准确的问题，本文在相机与激光雷达位置固定的情况下，针对多帧图像与点云进行了匹配运算，对N次结果进行求取平均值。对平移量的平均值求取公式为

$$\bar{t} = \frac{1}{N} \sum_{i=1}^N tvec_i$$

而对于旋转量的求取平均值，先将上文求得的旋转转为四元数 $rvec_i$ 形式，再对四元数求取平均值。

$$r = \frac{1}{N} \sum_{i=1}^N rvec_i$$

$$\bar{r} = \frac{r}{\|r\|}$$

通过对多帧激光雷达点云与相机进行三维点匹配得到的刚体变换求取平均值，能够有效的抑制因激光雷达的传感器误差而造成的标定误差。

3.3 本章小结

本章介绍了什么是激光雷达点云注册与运动畸变，并且提出了插值矫正运动畸变的方法：在对点云进行多帧融合时，利用点云传输的特性，将点云在时间上为84个Packet单独发布，同时线性插值得到这些Packet的时间戳上的编码器角度值，将这些Packet点云依据编码器的角度注册到参考坐标系中，并且将所有的Packet点云进行叠加注册后发布为新的点云。由于对每帧点云多进行了84次旋转角度的插值，从点云图案上看，本文提到的方法极大地改善了运动畸变对点云注册的影响，完成了矫正运动畸变的目的。同时，本章还介绍了一种利用特制的标定板在相机与激光雷达坐标系中寻找三维特征匹配点的方法来进行相机的标定，并详细介绍了如何利用三维匹配点来计算出两个点集所在的坐标系之间的刚体变换（Kabsch算法）。在后续的章节，本文将利用相机与激光雷达标定出来

的 $[R|t]$ 矩阵，对激光雷达与相机图像的数据进行融合处理，借以进行三维障碍物的分割与分类。

第4章 基于视觉与三维点云融合的三维障碍物检测方法

随着深度学习领域内的卷积神经网络在目标检测任务中大放异彩，基于视觉的目标检测的深度学习算法大行其道。在著名的大规模目标检测挑战赛ILSVRC^[?]中，基于深度学习的目标检测算法连续六年（2012-2017）取得了优越的表现。2017年，ILSVRC的夺冠深度学习网络SENet在目标检测问题上的错误率仅为2.25%，亦同时宣告了基于图像的目标检测任务基本被攻克。

然而，基于视觉的目标检测无法得到物体的三维位置信息，并且受光照影响严重。为了解决这些问题，本章后续部分将提出一种激光雷达与相机的多模态传感器融合技术，其能够在基于视觉的目标检测算法检测到目标时，能够根据激光雷达的点云得到障碍物的三维位置信息，同时根据激光雷达的点云信息反馈于目标检测任务的识别上，提升视觉在光照条件不够良好的情况下的检测准确率。

4.1 YOLO—一种实时目标检测网络

在介绍相机图像与激光雷达点云的融合之前，本文将先介绍本文使用的视觉目标检测算法。本文使用YOLO(You Only Look Once)^[?]算法作为视觉图像上的目标检测算法。

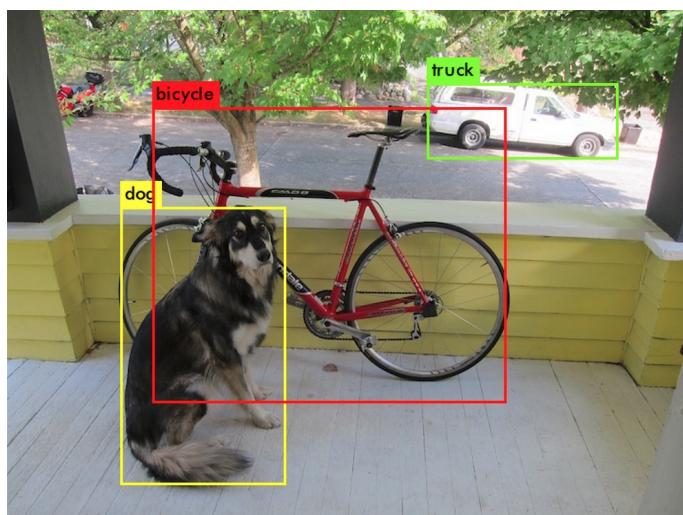


图 4-1 YOLO—一种实时目标检测网络

早期的目标检测算法通过提取图像的一些特征（例如Haar、SIFT、HOG等），运用DPM（Deformable Parts Model）模型，并且通过滑动窗口来预测具有较高得分的区域的策略来进行目标检测。这种传统的方法不仅相当耗时，而且检测准确率也不是很高。

随后出现了基于object proposal策略的方法，相比于滑动窗口这种类似于穷举的方法，该方法大大减少了运算量，同时定位精度也得到了很大的提高。结合13年兴起的卷积神经网络后，Object detection的性能得到了质的飞跃。

然而，诸如R-CNN、Faster R-CNN这样的网络，因为候选区域较多、网络运算量较大，因而很难在GPU性能不够强劲的移动机器人场景中做到实时检测。而本文采用的YOLO网络将目标检测问题转化为一个回归问题。给定图像 I ，YOLO网络直接在图像的多个位置上回归出识别目标的边界框（bounding box）以及其类别。

YOLO没有选择滑动窗口或者提取proposal的策略来训练网络，而是直接将整张图作为网络的输入，既极大地提升了运算速度，亦很好的区分了图像的前景与背景区域，而使用proposal策略的Fast R-CNN则经常将背景区域误识别为目標区域。

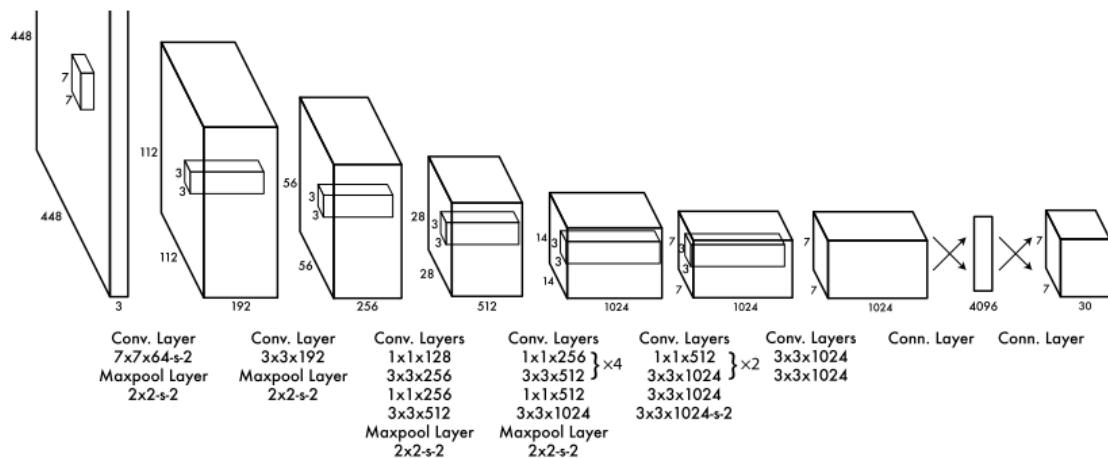


图 4-2 YOLO 网络结构

图4-2为YOLO的网络结构，从图中可以看出，该目标检测网络拥有24层卷积层，随后用两层全连接层对结果进行回归输出。

由于其简介的网络结构设计，使得其在实时性能上表现卓越。YOLO能够在每秒推断（inference）45帧的情况下仍能够保证和Faster R-CNN同样的准确率。基

于其良好的实时性与较为优秀的目标检测准确率，本文将使用YOLO作为视觉与三维点云融合的目标检测基准方法，并利用激光雷达的点云信息为其检测结果提供三维位置信息并提高准确率。

4.2 基于YOLO的视觉、三维点云结合的三维障碍物检测

本章所述的传感器设置如图4-3所示。其中激光雷达坐标系到相机坐标系的刚体变换已由第二章介绍的标定方法得到。在介绍如何将激光雷达点云投影到相机图像上之前，本文将先介绍相机的成像原理，这是点云投影的理论基础。

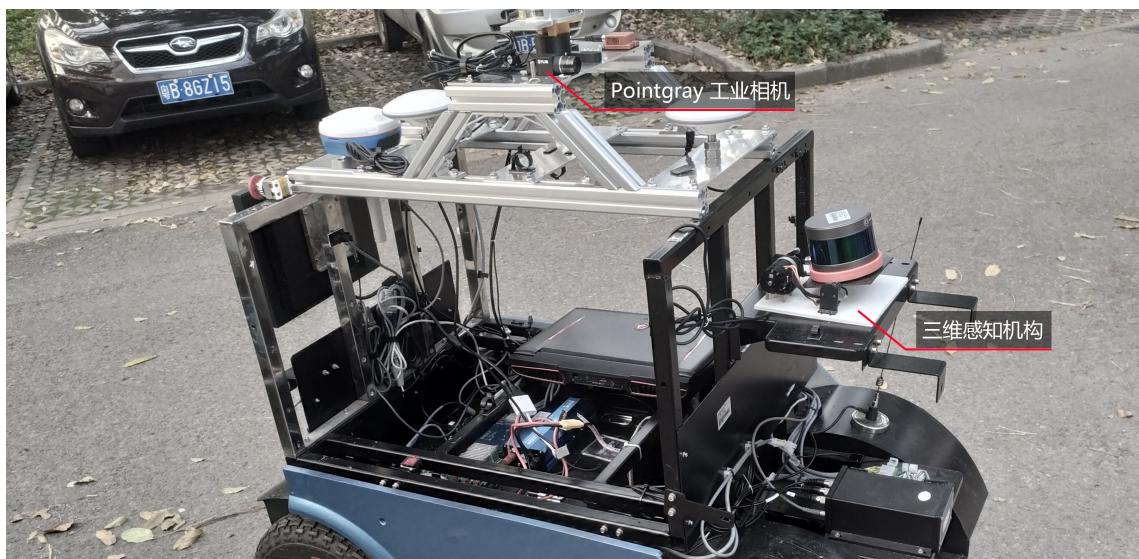


图 4-3 传感器设置

4.2.1 相机成像原理

所谓的相机成像，就是相机将三维空间中的坐标映射到二维图像平面的过程。这一映射可以有许多数学模型去解释，最简单也最为常用的就是针孔成像模型，如图4-4所示。现在对该模型进行建模。设图中 $O - x - y - z$ 为相机坐标系。通常认为相机坐标系 Z 轴指向相机的前方， X 轴指向右方， Y 轴指向下方。设真实空间中有一三维点 P ，经过相机的小孔 O 成像后，成像点落在相机的成像平面 P' 处，设 P 点的坐标为 $[X, Y, Z]^T$ ， P' 点的坐标为 $[X', Y', Z']$ ，并且设相机的成像平面到模型中的针孔的距离为 f （焦距），则根据三角形的相似关系，有

$$\frac{Z}{f} = -\frac{X}{X'} = -\frac{Y}{Y'} \quad (4-1)$$

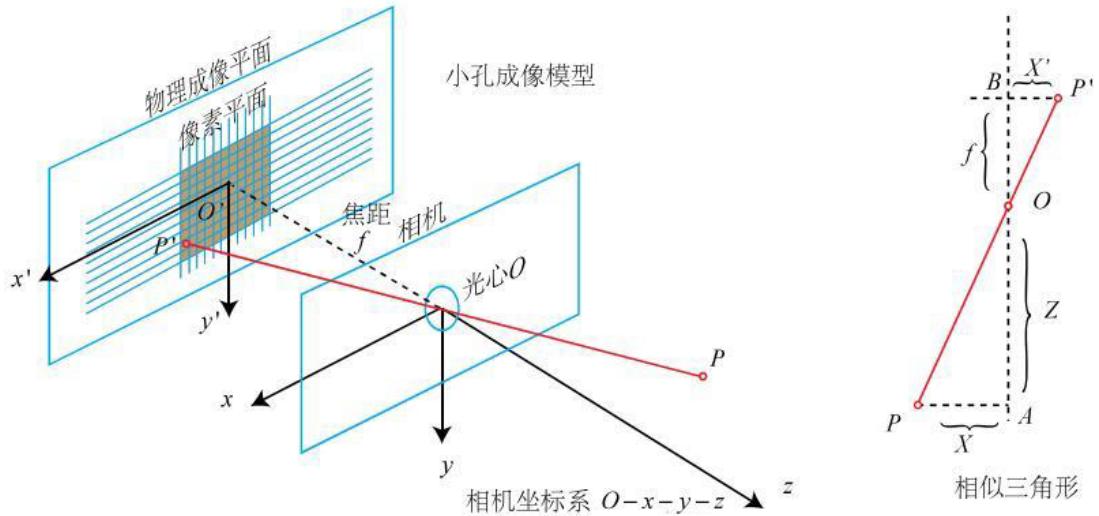


图 4-4 相机针孔成像原理

整理得

$$\begin{cases} X' = f \frac{X}{Z} \\ Y' = f \frac{Y}{Z} \end{cases} \quad (4-2)$$

实际上，在相机中最后得到的是一个个的像素，设在相机成像平面上存在一个像素平面 $O - u - v$ ， P' 在像素平面的坐标为 $[u, v]^T$ 。像素坐标系通常定义其原点 O' 在图像的左上角，其 u 轴与 v 轴的方向与相机坐标系中的 X 轴与 Y 轴相同。像素坐标系相对于相机坐标系，相差了一个平移与缩放。假设相机坐标系在 u 轴上缩放了 α 倍，在 v 轴上缩放了 β 倍，同时像素坐标系相对于相机坐标系的原点平移了 $[c_x, c_y]^T$ 。则 P' 在成像平面上的坐标与其像素坐标 $[u, v]^T$ 之间的关系为

$$\begin{cases} u = \alpha X' + c_x \\ v = \beta Y' + c_y \end{cases} \quad (4-3)$$

将上式代入式 4-2 中，并设 $\alpha f = f_x$ ， $\beta f = f_y$ ，则有

$$\begin{cases} u = f_x \frac{X}{Z} + c_x \\ v = f_y \frac{Y}{Z} + c_y \end{cases} \quad (4-4)$$

将上式写为矩阵的形式，则有

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{z} \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \triangleq \frac{1}{Z} KP \quad (4-5)$$

其中矩阵 $\begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix}$ 称之为相机的内参，通常，相机的标定就是为了得出相机的内参矩阵。本文在融合激光与相机之前已通过ROS的标定程序得到了相机的内参 K 。

4.2.2 激光雷达点云的投影

上式中， P 为相机坐标系内的一点，所以在根据第二章得到的标定方法得出激光雷达与相机坐标系的变换矩阵 T 后，先将激光雷达点云转换到相机坐标系中，再通过内参矩阵投影到图像上，即

$$P_{uv} = \frac{1}{Z} KTP_{LiDAR} \quad (4-6)$$

其中 P_{LiDAR} 表示激光雷达坐标系中的点云中的三维点。



图 4-5 激光雷达点云的投影

本文利用前文提到的三维感知机构进行了多帧点云的注册融合，并将融合后的点云投影到了相机的图像上，点云投影效果如图4-5所示。其中投影点的颜色

越深，代表其距离相机的位置越近；颜色越浅，代表离相机的距离越远。在将激光雷达点云投影到图像之后，便可以利用YOLO的检测结果对点云进行分割与分类。

4.2.3 点云前景与背景的分割

本文首先利用YOLO对相机图像进行了目标检测的推断。YOLO的检测效果如图4-6所示，其输出为多个边界框与识别的物体的类别，以及YOLO对推断结果的置信度（confidence）。图4-6为设置YOLO检测的置信度阈值为0.8时输出的结果（即不输出置信度低于0.8的检测结果）。



图 4-6 激光雷达点云的投影

将点云按照上文投影到图像中，则点云中的每个点 P' 都有一个唯一的像素坐标 $[u, v]^T$ 与其对应。剔除像素坐标不在YOLO检测的边界框内的点，便得到了基于YOLO检测结果的点云分割与分类结果，如图4-7(a)所示。

从图4-7(a)中可以看出，尽管按照上述结果的确将人与汽车的点云分割了出来，然而由于YOLO得出的边界框内除了识别的目标，还有一些背景物体，体现

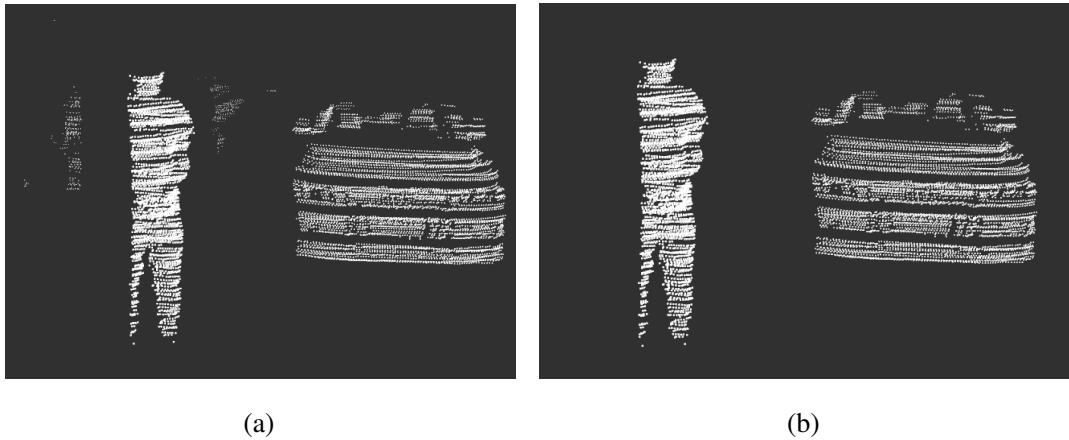


图 4-7 点云前景与背景的分割(a)原始点云;(b)Kmeans分类后的结果

在点云中就是，除了表示人与轿车的点云，还有一些人与汽车后的灌木的点云，其在图像上的投影也在候选框里。这些点云如果不加以去除，会对目标物体的三维位置的确定造成较大的负面影响。

考虑到目标点云与背景点云转换到相机坐标系后，在Z轴方向上有较大的距离，本文采用K-means算法^[?]，对投影后在同一边界框内的点云进行聚类分割。

K-means算法的目的是，把 n 个点划分到 K 个聚类中，使得每个点都属于离它最近的均值（称之为聚类中心）对应的聚类，以之作为聚类标准。其算法流程为：

- 1.假设分为K类；
- 2.任意随机出K个点，认为该K个点为聚类的中心点；
- 3.遍历每一个点 x ，计算其与上一步得出的K个点的欧式距离 $D(x)$ ，将该点归为使得 $D(x)$ 最小的聚类中；
- 4.计算每个聚类内所有点的平均位置，认为其为新的聚类中心；
- 5.重复3和4，直到 K 个聚类中心被选出来；
- 6.利用得到的新的K个点，继续重复3-5的步骤，直到每个点的分类结果不变，或者循环达到迭代次数上限。

对于图4-7(a)中的每个物体边界框内的点云，假设其可以被分为两类：前景与背景。设原始三维点集为 P_3 ，将所有点映射到相机坐标系的Z轴上，成为一个新的一维点集 P_1 ，并对 P_1 进行K-means聚类，其中 $K = 2$ 。则聚类得到的两类点中，

中心点Z轴坐标较小的即为前景，也就是YOLO检测的物体的点云。图4-7(b)为K-means聚类提取得到的结果。从图中可知，采用K-means算法较好的将前景与背景分割开来，剔除了无关的背景点，只保留了与YOLO检测到的目标有关的点。

4.2.4 目标三维坐标的计算与检测结果的优化

在得到了与YOLO的检测结果相关的目标物体的点云后，物体的三维坐标可以由求点集的中心点坐标而得。同时，为了更好的表示三维物体的检测结果，本文还将检测得到的物体用长方体包围盒去拟合，如图4-9所示。包围盒的顶点由点集的 $X_{min}, Y_{min}, Z_{min}$ 以及 $X_{max}, Y_{max}, Z_{max}$ 决定，其中 X_{min} 表示点集中X轴坐标最小的点的X坐标， X_{max} 表示点集中X轴坐标最大的点的X坐标。

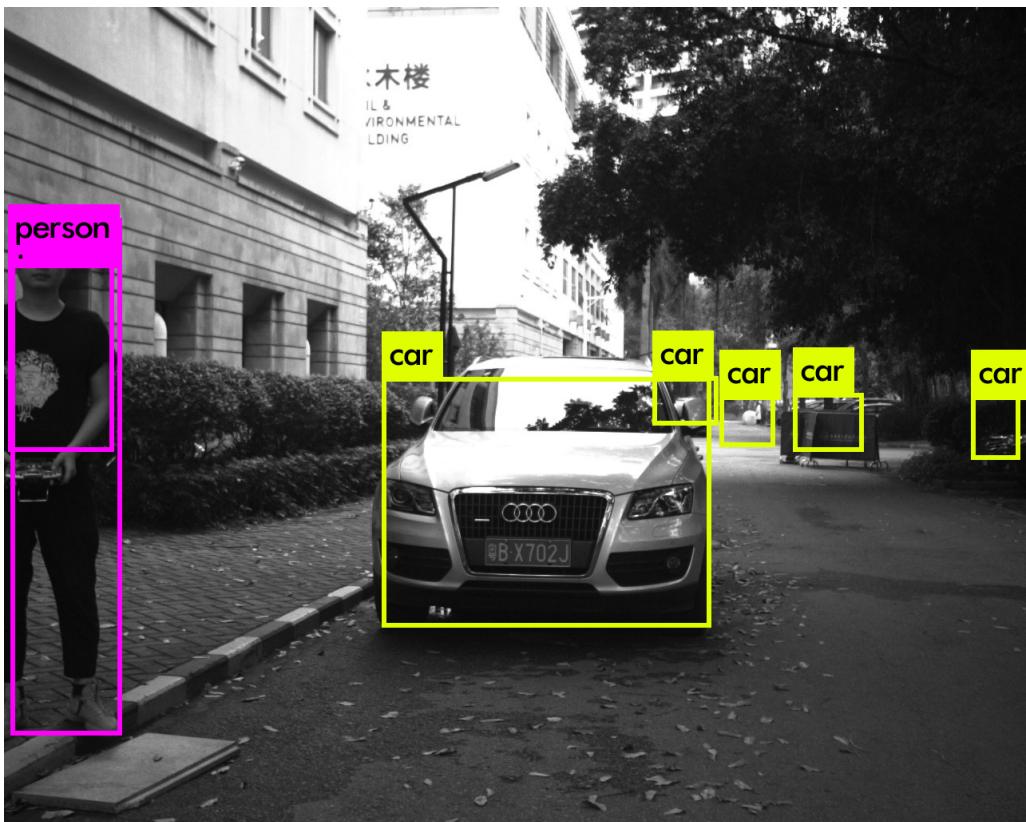


图 4-8 YOLO在低置信度阈值下的误检测

前文提到，图4-6中的结果是将YOLO检测结果的置信度阈值设为0.8后的检测结果。而将YOLO检测结果的阈值设为0.4后，YOLO就出现了许多误检测，如图4-8所示。由于光照等情况的影响，很多时候YOLO的检测结果置信度都不会高

于0.8，而降低置信度阈值又会造成误检测的结果增多。本文提出了一种在YOLO低置信度阈值下，融合激光雷达点云位置信息来祛除误检测结果的方法。

从图4-8中可以看出，大部分误检测的区域，其边界框对应的三维空间区域可能没有点云，或者其三维位置信息与边界框面积不相符合。为了祛除误检测结果，本文首先假设已知每个待检测的目标物体的最小投影矩形。举例来说，对图4-8来说，认为汽车在相机坐标系中的最小投影矩形为 $2.4m \times 1.8m$ ，即对于轿车而言，至少有 $2.4m \times 1.8m$ 的面积在相机视野范围内，那么，祛除误检测的策略流程可以表述为：

- 1.如果YOLO检测得到的边界框内没有点云，则认为发生了误检测。
- 2.如果YOLO检测得到的边界框内有点云，则计算点云中所有点的平均坐标

$$X_{mean}, Y_{mean}, Z_{mean}.$$

- 3.在相机坐标系中构建长方形平面，其顶点坐标为

$$\begin{aligned} & (X_{mean} - X_{hypo}, Y_{mean} - Y_{hypo}, Z_{mean}) \\ & (X_{mean} + X_{hypo}, Y_{mean} - Y_{hypo}, Z_{mean}) \\ & (X_{mean} - X_{hypo}, Y_{mean} + Y_{hypo}, Z_{mean}) \\ & (X_{mean} + X_{hypo}, Y_{mean} + Y_{hypo}, Z_{mean}) \end{aligned} \quad (4-7)$$

其中， X_{hypo}, Y_{hypo} 为上文提到的最小投影矩阵的长与宽。

- 4.将该长方形平面通过内参矩阵投影到相机图像中，得到一个估计的长方形框选面积 S_{hypo} 。
- 5.将 S_{hypo} 与YOLO检测出的边界框的面积 S_{yolo} 做比较，认为不满足约束条件

$$0.5 \times S_{hypo} \leq S_{yolo} \leq 1.5 \times S_{hypo} \quad (4-8)$$

的检测结果为误检测。

经过该点云图像融合祛除误检测策略后，三维物体检测结果如图4-9所示。其中误检测的部分用红色矩形标出，而正确的检测部分利用上文提出的长方体包围盒表示出来。可以看出，通过该策略能够有效的排除YOLO在低置信度阈值下的误检测，并且利用激光雷达的信息，能够很好的计算出检测的目标的三维位置。

4.3 本章小结

本章介绍了YOLO，一种实时的视觉目标检测方法，该方法最大的优点在于其实时性，能够在较高的识别准确率下，达到每秒推断45帧的速度。并且之后融

合了相机与激光雷达的信息，对三维障碍物进行了分割与分类。首先将激光雷达的点云信息投影到图像上，利用YOLO的推断结果，首先对激光雷达点云做了前景的提取与分割；随后，根据点云信息，计算得出了检测的目标的三维位置信息，并计算出物体的三维包围盒；最后，针对YOLO在低置信度阈值下容易出现误检测的问题，本文利用点云信息来验证YOLO的推断结果，并且甄别与祛除了误检测的输出。从结果中可以看出，融合视觉与三维点云的信息，能够较好的对三维物体进行检测与识别，为无人驾驶技术中的环境感知与路径规划提供了更多的信息。

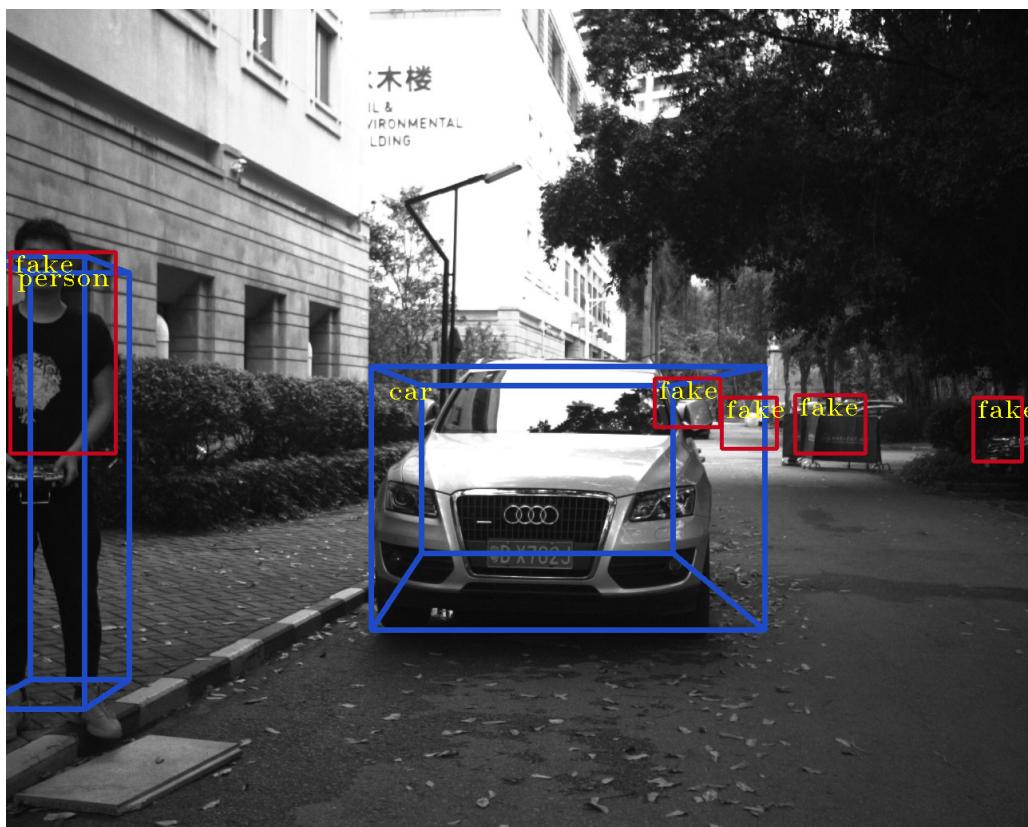


图 4-9 enhanced YOLO

第5章 实验验证与结果分析

本章将对文中提出的三维感知机构的进行仿真，对融合后的点云的注册融合效果进行了实验分析，并且融合了Odometry的信息。

5.1 Gazebo下的三维感知机构仿真

Gazebo是一个功能强大的三维物理仿真平台，具备强大的物理引擎、高质量的图形渲染、方便的编程与图形接口，最重要的还有其具备开源免费的特性。Gazebo支持显示逼真的三维环境，包括光线、纹理、影子等。它还支持传感器数据的仿真，同时可以仿真传感器噪声。

本文利用Gazebo对三维感知机构的运动以及点云融合进行了仿真，如图5-1(a)以及图5-1(b)所示。

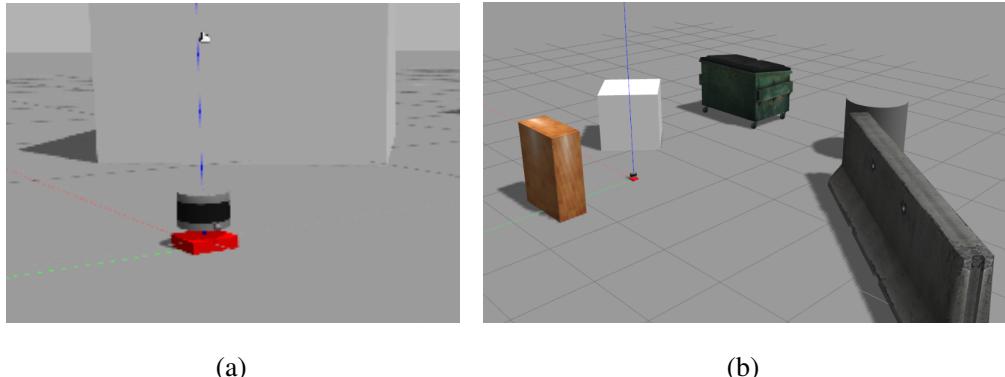


图 5-1 Gazebo仿真(a)三维感知机构的模型;(b)仿真环境

三维感知机构的Gazebo模型如图5-1(a)所示。由于在加上了ROS control的控制器后，利用单个圆柱形的link连接雷达，控制link做旋转运动就能够使得激光雷达在航向角上做来回的正弦运动，而进行Gazebo仿真的目的主要是为了仿真出激光雷达在往复运动情况下的点云注册融合情况，因此本文没有在Gazebo中设计曲柄摇杆机构。在仿真实验中，激光雷达传感器使用的是Velodyne 的VLP-16型激光雷

达。Gazebo支持传感器数据的仿真，按照第三章的注册融合策略融合后的点云如图所示。

5.2 三维感知机构结合里程计信息构建三维地图

前文提出的三维感知机构，其实验场景都是在机构位置静止不变的情况下得到的。而当机构架设在小车上时，由于小车相对于世界坐标系有一个运动，如果仍然采用之前的融合策略进行点云的输出，则输出的点云会有小车运动方向上的畸变。

因此，在运动的小车上进行三维感知时，需要结合里程计的信息，计算出机构在小车运动方向上的位移，借此消除点云在小车运动方向上的畸变。同时，可以根据小车自身坐标系到里程计坐标系的变换，利用该三维感知机构构建基于小车里程计坐标系的三维地图。

本文结合图4-3所示的传感器设置，利用小车的里程计对小车行驶过程中的点云进行了采集、融合与构建了三维地图。本文利用了ROS的TF坐标变换^[?]的设计，在结合里程计信息的同时，避免了繁琐的插值运算。

ROS的TF是一种这样的设计：TF是一个让使用者随时间跟踪多个坐标系的功能包。其数据结构类型为树状结构，能够根据时间缓冲并维护多个坐标系之间的坐标变换关系，可以帮助使用者在任意的时间点请求任意的坐标系之间的变换。

在本章的实现中，首先认为机构自身的坐标系为*baselink*，而激光雷达所在的坐标系为*lidar*，根据磁编码器返回的角度 α ，发布*baselink*坐标系到*lidar*坐标系的变换，认为其只有绕y轴旋转，即偏航角的变化，而机构的往复运动没有造成两个坐标系之间的位移，故其欧式变换矩阵为

$$\begin{bmatrix} \cos \alpha & 0 & \sin \alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \alpha & 0 & \cos \alpha & 0 \end{bmatrix}$$

随后，根据小车底层的编码器的信息，发布*odom*坐标系到*baselink*坐标系的变换。该欧式变换的旋转与位移皆通过小车的编码器与阿克曼转角的角度积分得到。

那么在TF树中，*odom*坐标系的子节点为*baselink*坐标系，*baselink*坐标系的子节点为*lidar*坐标系。则当激光雷达点云产生时，可以根据激光雷达的产生的点云信息的时间戳 t 来请求*odom*坐标系到*baselink*坐标系之间的欧式变换。尽管有可

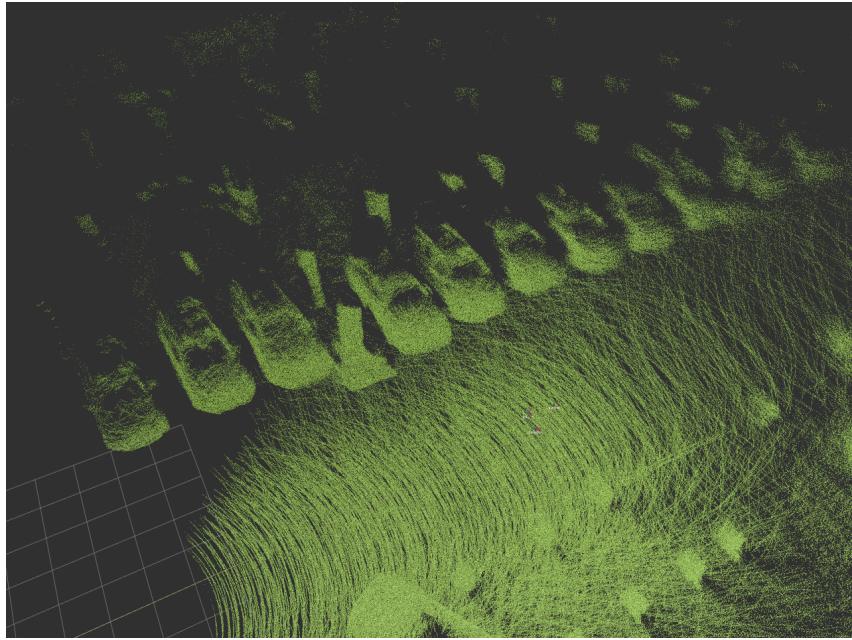


图 5-2 结合里程计生成的三维地图

能在时间戳 t 上没有发布 $odom$ 到 $baselink$ 、 $baselink$ 到 $lidar$ 坐标系的欧式变换，但是ROS的TF可以利用在时间戳 t 前后时刻已发布的欧式变换来进行插值得出 t 时刻的变换矩阵。有了 t 时刻的变换矩阵，可以很容易的将激光雷达坐标系下的点云变换到世界坐标系中。将每帧点云在世界坐标系中进行注册，则可得基于三维感知机构生成的三维点云地图，如图5-2所示。

5.3 基于点云投影到深度图像上的物体分割

在得到了基于三维感知机构融合与注册的点云后，一个重要的步骤就是进行点云的分割，进而求得点云中可能为障碍物的部分。本章节的实验环境如图5-3所示。

在该实验中，首先进行三维感知传感器机构对点云的融合，融合结果如图5-4(a)所示。关于点云注册融合的细节，本文已经在第三章中详细阐述，此处便不再赘述。

5.3.1 地面点的去除

有关点云的聚类，实际上就是指将三维空间点中的相近的点认为是同一个物体，将其归为同一类。而从点云图像中可以看出，地面点构成了点云的绝大部分。



图 5-3 实验环境

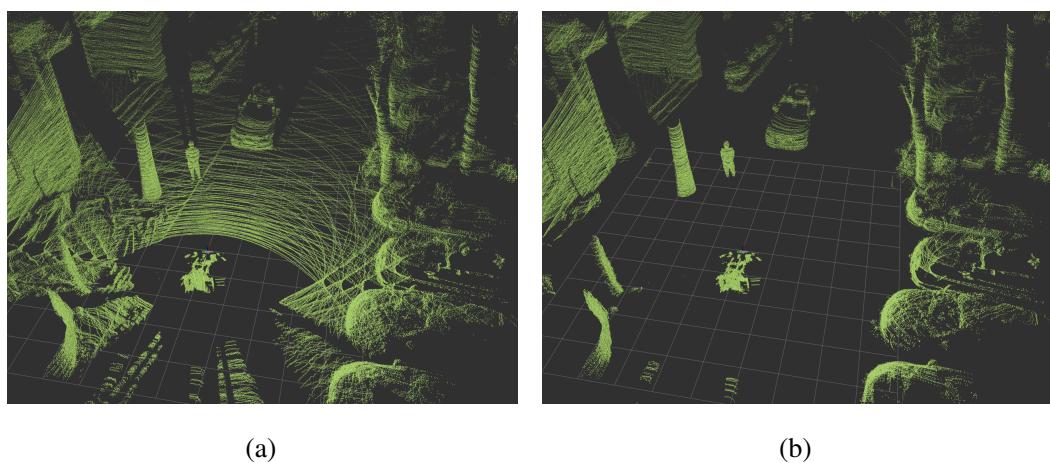


图 5-4 多帧融合后的点云(a)原始点云;(b)去除地面点后的点云

在聚类的过程中，地面点会对聚类的结果造成较大的影响，体现在地面点将两个不同物体的点云连接了起来，从而使得聚类算法会将两个不同的物体归为一类。本文采用文献^[?]提出的算法，首先对地面进行平面拟合，求出近似平面的方程，随后根据方程将平面方程以下的点去除。

在去除地面点之前，该算法首先提出了两个假设：

- 1.认为地面可以被近似为一个平面，即认为环境中地面是较为平整而不是有曲率的。
- 2.认为地面点永远是点云中高度最低的一些点。

算法流程如Algorithm 2所示。该算法使用最低点作为代表平面的点（lowest point representative）。首先，该算法先提取地面点的“种子（seed）”点，即认为最低的 N_{LPR} 个点中，高度为平均值再加一个阈值为 Th_{seeds} 的点为地面点。这个阈值有效的预防了传感器噪声的影响，即避免了因传感器噪声而导致的过低的点对地面的平面拟合造成影响。

为了估计地面的方程，该算法使用了一个简单的线性方程：

$$\begin{aligned} ax + by + cz + d &= 0 \\ n^T x &= -d \end{aligned} \tag{5-1}$$

其中， $n = [a, b, c]^T$ ， $X = [x, y, z]^T$ ，该方程首先求得关于种子点的协方差矩阵，通过协方差矩阵求得平面的法向量 n 。协方差矩阵 C 通过下式

$$C = \sum_{i=1:|S|} (s_i - \hat{s})(s_i - \hat{s})^T \tag{5-2}$$

而得。其中 $\hat{s} \in R^3$ 是所有 $s_i \in S$ 的平均值。

协方差矩阵 C 表征了种子点的离散程度，通过对对其进行奇异值分解（singular value decomposition）可得其奇异向量（singular vectors），奇异向量描述了其在三个方向上的离散程度。考虑到地面较为平坦，其种子点在竖直方向上的离散程度比较小，则与地面垂直的法向量 n 即为三个奇异向量中模最小的一个。

在得到法向量 n 之后，在式5-1中，令 $x = \hat{s}$ ，求得 d 。如此便得到了对地面进行拟合的平面方程。对于点云中的每个点 $p(x, y, z)$ ，求 $n^T x + d$ 的值，若大于0，则在平面之上（为非地面点）；若小于0，则在平面之下（为地面点）。

5.3.2 点云到深度图像的投影

正常的16线激光雷达点云每帧拥有三万个三维点，而由于本文提到的三维感知机构融合注册了多帧点云，其发布的点云每帧高达几十万个三维点。如此巨大的数据规模，如果采用常规的三维点云聚类方法，其每帧耗时将相当可观（在三维点云上的聚类算法通常时间复杂度在 $O(n \log(n))$ 以上）。因此需要一些对数据进行降采样的方法来提高算法的效率。

第二种方法是将去除地面点后的三维点云投影到地面的栅格平面上，这种方法也称为生成点云的鸟瞰图（bird's eye view）。随后在二维图像上进行物体的分割。这种方法运算速度很快，适合实施运算。然而这种方法有可能不能充分分割障碍物，如果多个物体彼此比较接近，它们有可能被认为是同一个物体。这取决于给地面划分栅格时栅格的大小，所以在不同的环境下，有可能要调整不同的栅格大小来改进算法。

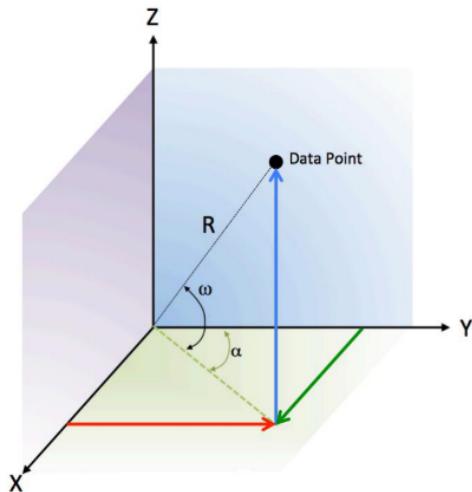


图 5-5 三维点的球坐标表示

而本文参考文献^[?]提出的方法，将点云投影到深度图像（Range Image）上，并且不需要在深度图像中提取特征，而是通过一种图搜索的算法将物体进行分割与定位。也因此，其时间复杂度为 $O(n)$ 级别，能够满足实时性的要求。

将点云投影为深度图像，首先要将点云中的点的表示由笛卡尔坐标系转换为球坐标系，如图5-5所示。球坐标系中的三维点 $p = (\alpha, \omega, r)$ ，其中 α 与 β 的表示如图所示， r 为三维点到坐标系原点的距离。则显然，点的球坐标系与笛卡尔坐标系坐

标转换的关系为

$$\begin{cases} r = \sqrt{x^2 + y^2 + z^2} \\ \omega = \arcsin \frac{z}{r} \\ \alpha = \arccos \frac{y}{r \cos \omega} \end{cases} \quad (5-3)$$

求得三维点的球坐标后，就要根据三维点的 α 与 ω 来进行到深度图像的投影。假设深度图像的长与宽为 w 与 h ，并且空间中有一三维点 $p_{3d} = (\alpha, \omega, r)$ ，首先设融合后的点云中的点的最大 α 角为 α_{max} ，最大 ω 角为 ω_{max} ，则三维点 p_{3d} 映射到二维点 $p_{2d} = (u, v)$ 的关系为

$$\begin{cases} \alpha_{percol} = 2\alpha_{max}/w \\ \omega_{perrow} = 2\omega_{max}/h \\ u = (\alpha/\alpha_{percol} + w/2) \mod w \\ v = -\omega/\omega_{perrow} + h/2 \end{cases} \quad (5-4)$$

其中 α_{percol} 与 ω_{perrow} 是深度图像中每行与每列代表的角度，而有关 u, v 的变换是为了让激光雷达坐标系中正前方的点云能够投影到深度图像的中心。经过投影后得到的深度图像如图5-6所示，其中像素点的像素值为三维点的 x 坐标值，像素值越大，颜色越亮，表示该像素所表征的三维点离相机越远。



图 5-6 由点云投影得到的深度图像

5.3.3 基于深度图像的物体分割

投影后的深度图像相当于将具有相同 α, ω 而 r 不同的点用同一个位置的像素点来表示，因而相当于通过损失有限的信息对点云信息进行了压缩。在实际无人车的行驶情况中，由于当 α, ω 相同时，可以先关注 r 比较小的点，这些点表示同一方

向上近处的物体，这对无人驾驶时的路径规划是十分重要的，因而这样的信息损失是完全可以接受的。

对于深度图像上的物体分割，本文采用广度优先搜索（Breadth First Search, BFS）的思想，对深度图像中每个像素值不为0的点，搜索其邻域内的点，若其像素值（即实际的距离）与当前像素值之差不超过某个阈值，则认为其为同一类物体。算法流程如Algorithm 3所示。

在实际的实现中， $Thres$ 取值为0.5，分割的结果如图5-7所示。本文将每个不同的聚类的物体标记为不同的颜色，从图中可以看出，这种方法很好的将空间位置不同的物体分割了出来。

在深度图像上将物体分割后，一个重要的步骤就是根据图像中物体的像素坐标以及深度值反推出物体在世界坐标系中的坐标。假设该物体所有像素的平均坐标为 $p(\bar{u}, \bar{v})$ ，其平均深度通过求所有像素的平均值而得，设为 \bar{r} 。则可以根据三维点云投影深度图的算法倒推得其三维坐标：

$$\begin{cases} \alpha = (\bar{u} - w/2) * \alpha_{percol} \\ \omega = -(\bar{v} - h/2) * \omega_{perrow} \\ x = \bar{r} \cos(\alpha) \cos(\omega) \\ y = -\bar{r} \cos(\alpha) \sin(\omega) \\ z = \bar{r} \sin(\omega) \end{cases} \quad (5-5)$$



图 5-7 由点云投影得到的深度图像

由此，便实现了基于点云投影到深度图像上的实时物体分割与定位。

5.4 本章小结

Algorithm 2 ground plane fitting methodology for one segment of the point cloud

Output:

P_g : points belonging to ground surface

P_{ng} :points not belonging to ground surface

1: **Initialization:**

2: P : input point cloud

3: N_{iter} : number of iterations

4: N_{LPR} : number of points used to estimate the LPR

5: Th_{seeds} : threshold for points to be considered initial seeds

6: Th_{dist} : threshold distance from the plane

7: **Main Loop:**

8: $P_g = \text{ExtractInitialSeeds}(P, N_{LPR}, Th_{seeds})$;

9: **for** $i = 1 : N_{iter}$ **do**

10: model = EstimatePlane(P_g);

11: clear(P_g, P_{ng});

12: **for** $k = 1 : \|P\|$ **do**

13: **if** $model(p_k) < Th_{dist}$ **then**

14: $P_g \leftarrow p_k$;

15: **else**

16: $P_{ng} \leftarrow p_k$;

17: **ExtractInitialSeeds:**

18: $P_{sorted} = \text{SortOnHeight}(P)$;

19: $LPR = \text{Average}(P_{sorted}(1 : N_{LPR}))$;

20: **for** $K = 1 : \|P\|$ **do**

21: **if** $p_k.height < LPR.height + Th_{seeds}$ **then**

22: $seeds \leftarrow p_k$;
return $seeds$

Algorithm 3 Range Image Labelling

```
1: LabelRangeImage(R)
2: Label  $\leftarrow 1, L \leftarrow zeros(R_{rows} \times R_{cols})$ 
3: for  $r = 1 : R_{rows}$  do
4:   for  $c = 1 : R_{cols}$  do
5:     if  $L(r, c) == 0$  then
6:       LabelComponentBFS( $r, c, Label$ );
7:       Label = Label + 1;
8:     LabelComponentBFS(r, c, Label)
9:     queue.push( $\{r, c\}$ )
10:    while queue is not empty do
11:       $\{r, c\}$  = queue.top();
12:       $L(r, c) = Label$ 
13:      for  $\{r_n, c_n\} \in Neighbourhood\{r, c\}$  do
14:        if  $abs(R(r_n, c_n) - R(r, c)) < Thres$  then
15:          queue.push( $\{r_n, c_n\}$ )
queue.pop()
```

第6章 全文总结与后续工作展望

6.1 全文总结

本文以时域积分方程方法为研究背景，主要对求解时域积分方程的时间步进算法以及两层平面波快速算法进行了研究。

.....

6.2 后续工作展望

时域积分方程方法的研究近几年发展迅速，在本文研究工作的基础上，仍有以下方向值得进一步研究：

.....

致谢

致 谢

The Name of the Game

1.1 xxx

1.1.1 xxx

1.1.1.1 xxxx

1.2 xxx

1.2.1 xxx

1.2.1.1 xxxx

English words like ‘technology’ stem from a Greek root beginning with the letters $\tau\epsilon\chi\dots$; and this same Greek word means *art* as well as technology. Hence the name $\text{T}_{\text{E}}\text{X}$, which is an uppercase form of $\tau\epsilon\chi.\text{TeX}$ (actually $\text{T}_{\text{E}}\text{X}$), meaning of $\tau\epsilon\chi$

Insiders pronounce the χ of $\text{T}_{\text{E}}\text{X}$ as a Greek chi, not as an ‘x’, so that $\text{T}_{\text{E}}\text{X}$ rhymes with the word bleechhh. It’s the ‘ch’ sound in Scottish words like *loch* or German words like *ach*; it’s a Spanish ‘j’ and a Russian ‘kh’. When you say it correctly to your computer, the terminal may become slightly moist.

The purpose of this pronunciation exercise is to remind you that $\text{T}_{\text{E}}\text{X}$ is primarily concerned with high-quality technical manuscripts: Its emphasis is on art and technology, as in the underlying Greek word. If you merely want to produce a passably good document—something acceptable and basically readable but not really beautiful—a simpler system will usually suffice. With $\text{T}_{\text{E}}\text{X}$ the goal is to produce the *finest* quality; this requires more attention to detail, but you will not find it much harder to go the extra distance, and you’ll be able to take special pride in the finished product.

On the other hand, it’s important to notice another thing about $\text{T}_{\text{E}}\text{X}$ ’s name: The ‘E’ is out of kilter. This logo displaced ‘E’ is a reminder that $\text{T}_{\text{E}}\text{X}$ is about typesetting, and it distinguishes $\text{T}_{\text{E}}\text{X}$ from other system names. In fact, TEX (pronounced *tecks*) is the admirable *Text EXecutive* processor developed by Honeywell Information Systems.

Since these two system names are Bemer, Robert, see TEX, ASCII pronounced quite differently, they should also be spelled differently. The correct way to refer to T_EX in a computer file, or when using some other medium that doesn't allow lowering of the 'E', is to type '—TeX—'. Then there will be no confusion with similar names, and people will be primed to pronounce everything properly.

此名有诗意

1.1 xxx

1.1.1 xxx

1.1.1.1 xxxx

1.2 xxx

1.2.1 xxx

1.2.1.1 xxxx

英语单词“technology”来源于以字母 $\tau\epsilon\chi\dots$ 开头的希腊词根；并且这个希腊单词除了 technology 的意思外也有 art 的意思。因此，名称 **TEX** 是 $\tau\epsilon\chi$ 的大写格式。

在发音时，**TEX** 的 χ 的发音与希腊的 chi 一样，而不是 “x”，所以 **TEX** 与 bleechhh 押韵。“ch” 听起来象苏格兰单词中的 loch 或者德语单词中的 ach；它在西班牙语中是 “j”，在俄语中是 “kh”。当你对着计算机正确读出时，终端屏幕上可能有点雾。

这个发音练习是提醒你，**TEX** 主要处理的是高质量的专业书稿：它的重点在艺术和专业方面，就象希腊单词的含义一样。如果你仅仅想得到一个过得去——可读下去但不那么漂亮——的文书，那么简单的系统一般就够用了。使用 **TEX** 的目的是得到最好的质量；这就要在细节上花功夫，但是你不会认为它难到哪里去，并且你会为所完成的作品感到特别骄傲。

另一方面重要的是要注意到与 **TEX** 名称有关的另一件事：“E” 是错位的。这个偏移 “E”的标识提醒人们，**TEX** 与排版有关，并且把 **TEX** 从其它系统的名称区别开来。实际上，**TEX**(读音为 tecks)是 Honeywell Information Systems 的极好的 Text EXecutive 处理器。因为这两个系统的名称读音差别很大，所以它们的拼写也不同。在计算机中表明 **TEX** 文件的正确方法，或者当所用的方式无法降低 “E” 时，就要写作 “TeX”。这样，就与类似的名称不会产生混淆，并且为人们可以正确发音提供了条件。