摘　要

在机器人和无人机的世界中，正在稳步增长的一个趋势就是在用户允许的情况下对其进行“跟随”，这种趋势的玩法非常有趣，比如利用无人机搭配 GoPro 相机来拍摄自拍视频，不仅乐趣十足，而且相对于花重金聘请直升飞机和摄制组来说，成本要低廉许多。

除了无人机之外，如今有越来越多的陆地机器人也开始具备“跟随领路者”功能，比如 CaddyTrek 推出的电动高尔夫球童已经能够取代部分劳动力，所以能够追着你的影子到处跑的机器人已经指日可待了。

本文设计了一种基于 ROS 系统的跟随机器人，通过选择一个“显眼”的目标，机器人会随着目标的移动而移动。本系统最重要的就是跟随算法的实现，将近年来跟踪效果比较好的核相关滤波算法，成功的移植到 ROS 中，实现目标跟随的功能。

关键字 目标跟踪；核相关滤波；ROS 机器人操作系统；移动机器人

Abstract

In the robot and the UAV in the world, a trend is growing steadily is to "follow" the user permission, the trend of the play is very interesting, such as the use of UAV GoPro camera self timer video collocation, not only fun, but also to spend lots of money to hire a helicopter and crew., low cost to many.

In addition to the UAV, now there are more and more land robots have also started to "follow the leader" function, such as CaddyTrek launched the electric golf caddy has been able to replace part of the labor force, so to be able to chase your shadow around the robot has been just around the corner.

This paper describes the design of a follower robot based on the ROS system, by choosing a "conspicuous" target, the robot will move with the movement of the target. This system is the most important thing is to follow the implementation of the algorithm, kernel correlation filtering algorithm in recent years better tracking results, successfully transplanted to ROS, to achieve the goal follow the function.

Key words Target tracking Kernel correlation filter Robot Operating System Mobile robot

目　录

第 1 章　绪 论

1.1 引言

21 世纪是什么样的世纪？是物联网的世纪？是 VR 的世纪？也许吧，但我更相信 21 世纪是机器人的世纪。目前，我国已经步入经济转型的拐点区间，人口红利越来越难以支撑中国经济的发展和进步。在很多行业都已经开始了机器换人，生产工艺升级换代的步伐。工信部，发改委，财政部日前联合印发了《机器人产业规划（2016-2020）》。由此可见，机器人未来的政策空间和试场发展空间都是非常巨大的。一方面，发展工业机器人在满足我国制造业的转型升级，提质增效，实现“中国制造 2025”等方面具有极为重大的意义，是全面推进实施制造强国战略的重要一步。另一方面，从服务机器人来说，也要满足未来市场需求的增长。首先，这包括了基本生活需求，比如养老，助老，助残等。其次是国家安全需要，比如救灾，抢险，海底勘探，航天，国防。最后还有家庭服务和娱乐机器人，比如娱乐，儿童教育，智能家居应用等。

智能移动机器人是机器人研究的一个重要分支。它需要具备强健的“肢”，明亮的“眼”，灵巧的“嘴”以及聪慧的“脑”，这一切的实现实际上涉及诸多技术领域。目前移动机器人涉及的主要技术有运动控制，环境感知，导航与定位，最优路径规划等。

本文主要介绍了视觉目标跟踪技术，利用 Kinect 摄像头构建了可在较为复杂的环境中运行的智能移动机器人系统。

1.2 移动机器人发展史

至今，在工业制造领域，机器人学已经取得了巨大的成功。机械臂，机械灵巧手，焊接机器人以及喷漆机器人等构成了巨大的工业产值。但是，对于所有的这些成功应用，都有一个共同的缺点：缺乏机动性。固定的机械手被安装在固定的地方，其运动范围是有限的。相反，移动机器人能够行走，穿过整个制造工厂，灵活地在它最有效的地方施展才能。

60 年代后期，美国和苏联为完成月球探测计划，研制并应用了移动机器人。美国“探测者”3 号,其操作器在地面的遥控下，完成了在月球上挖沟和执行其他任务。苏联的“登月者”20 号在无人驾驶的情况下降落在月球表面，操作器在月球表面钻削岩石，并把土壤和岩石样品装进回收容器并送回地球。70 年代初期，日本早稻田大学研制出具有仿人功能的两足步行机器人。为适应原子能利用和海洋开发的需要，极限作业机器人和水下机器人也发展较快。国内的机器人发展较晚，但近年来在政府的支持下，也有了长足的进步。

1.3 ROS（Robot Operating System）简介

ROS 是一个分布式的系统，主要的的构成是节点（node）,每一个节点可以就看做一个单独的独立单元，它可以接收外部的消息（这个步骤叫订阅，subcribe）,也可以向外部发出自己的消息（这个步骤叫发布，publish）。服务是节点功能实现的一个典型的方式，也就是接收外部的消息，并完成对该消息的响应。这里有个经典的应用例子:一个提供加法运算服务的节点，定时查询消息，一旦接收到消息（俩个数值），立即返回加法运算的结果

（俩数之和），节点是一个单独的处理单元，这样会大大简化程序设计和代码量，每个节点在代码层看起来就是一个独立的执行文件，有一个 main()函数入口。

除此之外，节点之间的通讯是通过类似 TCP 通讯机制的通讯底层所实现的，所以整个 ROS 上的节点是否跑在同一运算平台上并不重要（不考虑运算能力），只要这些节点之间有高效的通讯通道，便可以让节点相对运算平台透明，这也是分布式系统的特点。还有一系列的仿真，调试工具，如 gazebo,rviz,rqt 等，更重要的是能够使用多种语言进行开发，如 C++,python 等。

1.4 视觉目标跟踪技术

目标跟踪是绝大多数视觉系统中不可或缺的环节。在二维视频跟踪算法中，基于目标颜色信息或基于目标运动信息等方法是常用的跟踪方法。从以往的研究中我们发现，大多数普通摄像头（彩色摄像头）下非基于背景建模的跟踪算法都极易受光照条件的影响。这是因为颜色变化在某种程度上是光学的色彩变化造成的。如基于体素和图像像素守恒假设的光流算法它也是假设一个物体的颜色在前后两帧没有巨大而明显的变化。

目标跟踪的实质实际上是模板匹配的问题，它的思想很简单，我们把要跟踪的目标保存好，然后在每一帧来临的时候，我们在整个图像中寻找与这个目标最相似的，我们就相信这个就是目标了。21 世纪以来，人们以现代滤波方法的概率形式来判断目标的匹配程度，常见的有卡尔曼滤波，扩展卡尔曼滤波，粒子滤波，相关滤波等。但模板匹配的前提是目标特征的选取，人们通常使用的图像中的目标特征有颜色特征，边缘特征以及光流特征等，第 3 章我们还会详细的描述。

1.5 本文主要研究内容

本文结合跟随机器人的主要技术难点，对目标跟踪的相关知识进行了比较深入的研究。

我们采用微软的 kinect 摄像机作为移动机器人的主要环境信息感知器，对 kinect 摄像机的结构，性能参数做了介绍。

为了实现视觉目标的跟踪，我们探讨了 KCF 算法，最终将 KCF 算法集成到 ROS 系统中，实现了跟随系统的设计。

第 2 章　机器人硬件系统

2.1 整体框架

为了能够实现目标跟踪，我们采用微软公司的 kinect 作为移动机器人的“眼睛”，用来获取跟踪目标与所在环境的信息。移动机器人采用 M-robot,是一群机器人爱好者开发的一款简易移动机器人。

系统如图 2.1 所示。

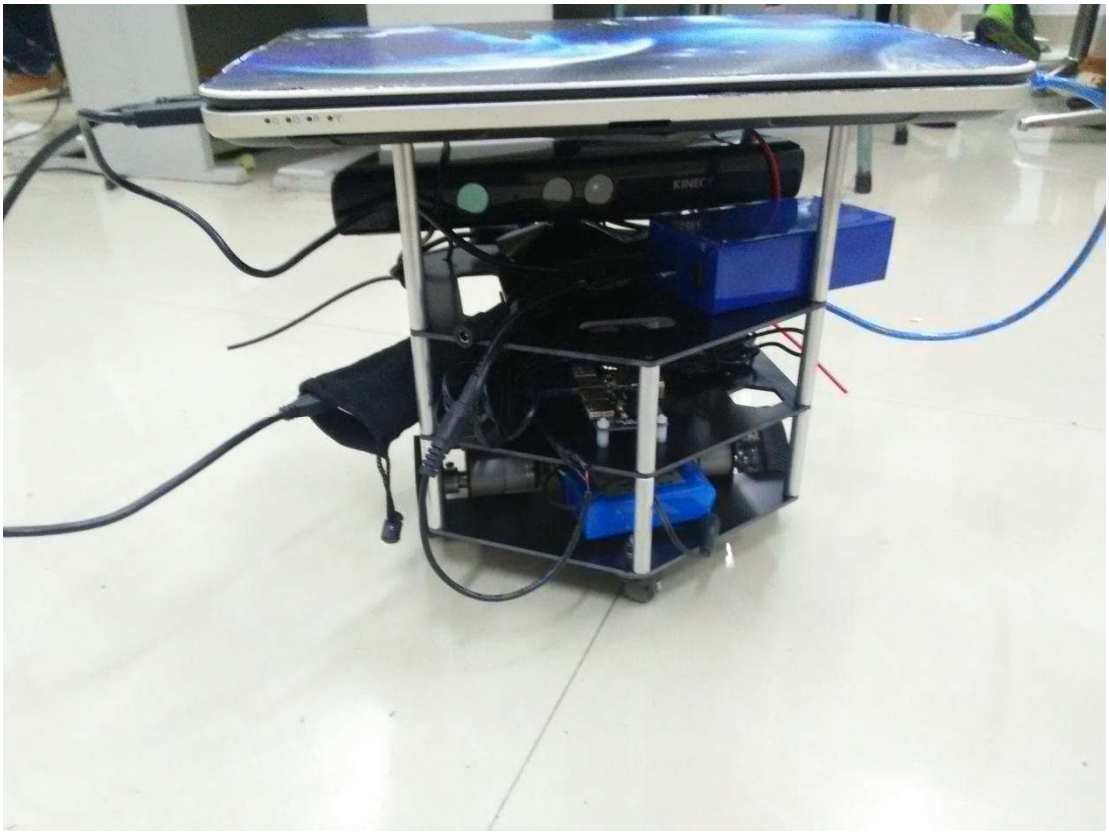

图 2.1 硬件平台

2.2 M-robot 机器人介绍

这个平台基于 ROS 开源系统构建，可实现室内建立地图和导航，能够作为 ROS 的学习平台，不能达到消费级产品的稳定性导航效果，可以在开源算法的基础上继续优化得到更好地效果。该移动机器人系统包括：

（1）底层采用自制的 Arduino2560 控制（2560+328 双控制芯片），遵循 iRobot 开源协议。

（2）拥有独立于 ROS 之外的接口：4 个超声波接口，2 个舵机控制接口，1 个 UART 接口，一个 I2C 接口。

（3）上位机采用笔记本电脑。

（4）系统参数：净重 3KG，负载能力 5KG，主动轮 2 个，全向轮 1 个，采用差分驱

动方式，精度可达厘米级别，速度为 0~1.0m/s,电源输入为 DC-12V，通信波特率为 57600/115200。

2.2.1 主控制器

主要资源有如下：

（1）主控芯片： Atmega 2560

（2）超声波控制芯片：Atmega328

（3）USB--TTL：FT232

（4）电源芯片：LM2596--5V

（5）板载接口：电源输入接口，5V 电压输出接口，开关，USB 串口，编码器输入接口，电机驱动控制接口，超声波接口，舵机接口，遥控接口，下载固件接口以及预留接口。

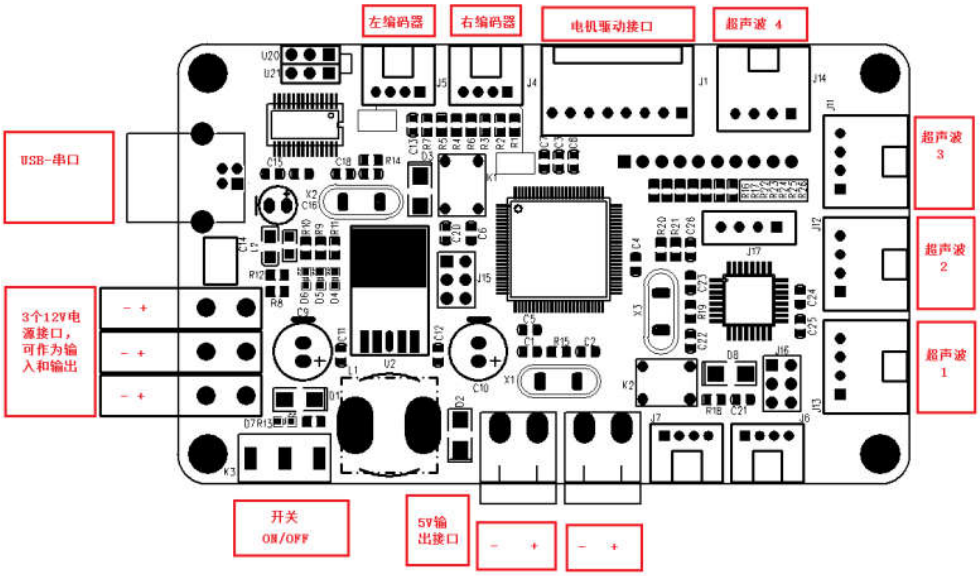

图 2.2 主控制器

2.2.2 电机驱动板介绍

电机驱动板适用于直流电机驱动，理论电流 100A，如果要控制大功率电机，需要加散热器。

板载接口有：

（1）电源输入接口，用于电机电源 7--12V。

（2）电机控制接口。

（3）总电源开关。

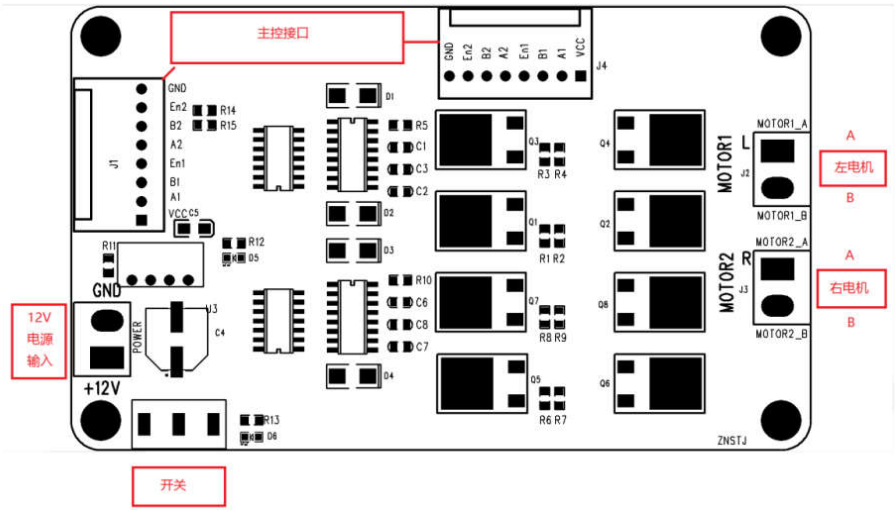

图 2.3 电机驱动板

2.2.3 电源板介绍

电源板适用于锂电池和蓄电池，板载资源有:

（1）电池输入接口，DC-5V
（2）电池充电口，DC-5V
（3）俩个 10A 保险丝
（4）电量指示模块接口
（5）总电源开关
（6）急停开关接口
（7）12V 主控接口，急停开关控制
（8）12V 输出，不受急停开关控制
（9）5V 输出

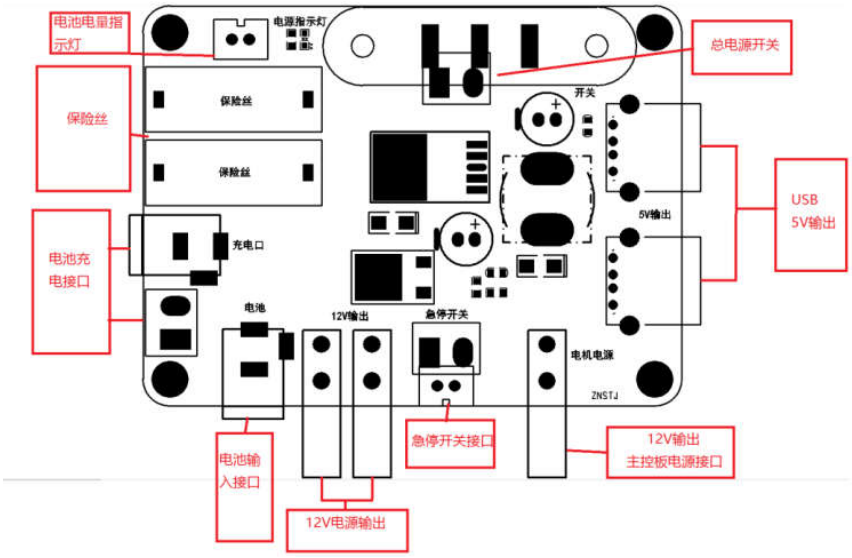

图 2.4 电源板

2.3 Kinect

Kinect for Xbox 360，简称 Kinect，是由微软开发，应用于 Xbox 360 主机的周边设备。它让玩家不需要手持或踩踏控制器，而是使用语音指令或手势来操作 Xbox360 的系统界面。它也能捕捉玩家全身上下的动作，用身体来进行游戏，带给玩家“免控制器的游戏与娱乐体验”。其在 2010 年 11 月 4 日于美国上市，建议售价 149 美金。Kinect 在销售前 60 天内，卖出八百万部，目前已经申请吉尼斯世界记录，成为全世界销售最快的消费性电子产品。

2012 年 2 月 1 日，微软正式发布面向 Windows 系统的 Kinect 版本“Kinect for Windows”，建议售价 249 美金。而在 2012 年晚些时候，微软还将发布面向“教育用户”的特别版 Kinect。

Kinect 有三个镜头，中间的镜头是 RGB 彩色摄影机，用来采集彩色图像。左右两边镜头则分别为红外线发射器和红外线 CMOS 摄影机所构成的 3D 结构光深度感应器，用来采集深度数据（场景中物体到摄像头的距离）。彩色摄像头最大支持 1280*960 分辨率成像，红外摄像头最大支持 640*480 成像。Kinect 还搭配了追焦技术，底座马达会随着对焦物体移动跟着转动。Kinect 也内建阵列式麦克风，由四个麦克风同时收音，比对后消除杂音，并通过其采集声音进行语音识别和声源定位。

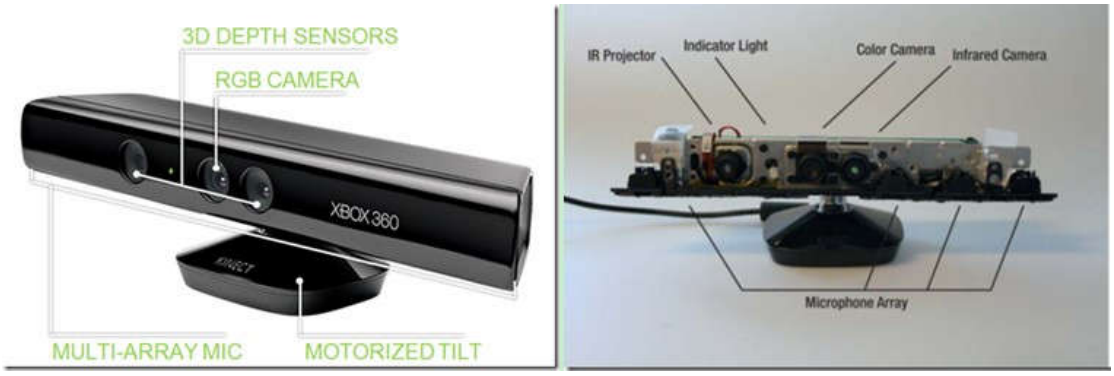

图 2.5 kinect

第 3 章　ROS 介绍

3.1 ROS 概览

随着机器人领域的快速发展和复杂化，代码的复用性和模块化的需求原来越强烈，而已有的开源机器人系统又不能很好的适应需求。2010 年 Willow Garage 公司发布了开源机器人操作系统 ROS（robot operating system），很快在机器人研究领域展开了学习和使用 ROS 的热潮。

ROS 系统是起源于 2007 年斯坦福大学人工智能实验室的项目与机器人技术公司 Willow Garage 的个人机器人项目（Personal Robots Program）之间的合作，2008 年之后就由 Willow Garage 来进行推动。随着 PR2 那些不可思议的表现，譬如叠衣服，插插座，做早饭，ROS 也得到越来越多的关注。Willow Garage 公司也表示希望借助开源的力量使 PR2 变成“全能”机器人。

PR2 价格高昂，2011 年零售价高达 40 万美元。PR2 现主要用于研究。PR2 有两条手臂，每条手臂七个关节，手臂末端是一个可以张合的钳子。PR2 依靠底部的四个轮子移动。在 PR2 的头部，胸部，肘部，钳子上安装有高分辨率摄像头，激光测距仪，惯性测量单元，触觉传感器等丰富的传感设备。在 PR2 的底部有两台 8 核的电脑作为机器人各硬件的控制和通讯中枢。两台电脑安装有 Ubuntu 和 ROS。

3.2 ROS 特点

ROS 是开源的，是用于机器人的一种后操作系统，或者说次级操作系统。它提供类似操作系统所提供的功能，包含硬件抽象描述、底层驱动程序管理、共用功能的执行、程序间的消息传递、程序发行包管理，它也提供一些工具程序和库用于获取、建立、编写和运行多机整合的程序。

图 3.1 ROS 产品

ROS 的首要设计目标是在机器人研发领域提高代码复用率。ROS 是一种分布式处理框架（又名 Nodes）。这使可执行文件能被单独设计，并且在运行时松散耦合。这些过程可以封装到数据包（Packages）和堆栈（Stacks）中，以便于共享和分发。ROS 还支持代码库的联合系统。使得协作亦能被分发。这种从文件系统级别到社区一级的设计让独立地决定发展和实施工作成为可能。上述所有功能都能由 ROS 的基础工具实现。

ROS 的运行架构是一种使用 ROS 通信模块实现模块间 P2P 的松耦合的网络连接的处理架构，它执行若干种类型的通讯，包括基于服务的同步 RPC（远程过程调用）通讯、基于 Topic 的异步数据流通讯，还有参数服务器上的数据存储。但是 ROS 本身并没有实时性。

ROS 的主要特点可以归纳为以下几条：

（1）点对点设计

一个使用 ROS 的系统包括一系列进程，这些进程存在于多个不同的主机并且在运行过程中通过端对端的拓扑结构进行联系。虽然基于中心服务器的那些软件框架也可以实现多进程和多主机的优势，但是在这些框架中，当各电脑通过不同的网络进行连接时，中心数据服务器就会发生问题。

ROS 的点对点设计以及服务和节点管理器等机制可以分散由计算机视觉和语音识别等功能带来的实时计算压力，能够适应多机器人遇到的挑战。

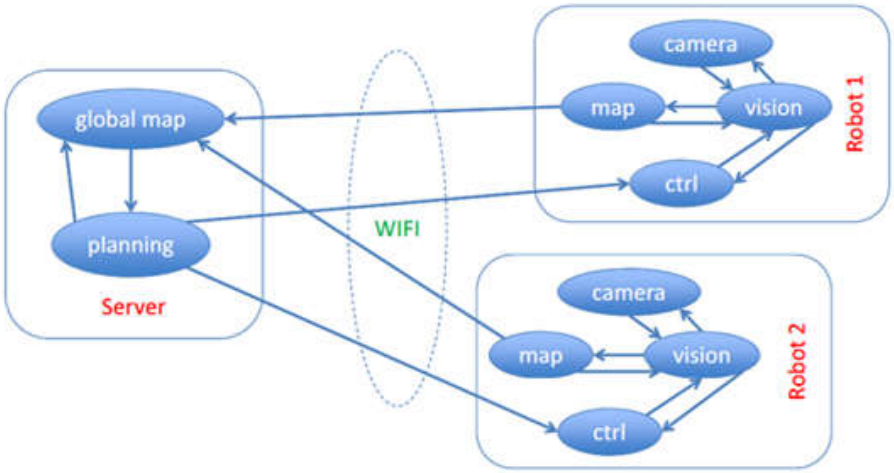

图 3.2 点对点设计

（2）多语言支持

在写代码的时候，许多编程者会比较偏向某一些编程语言。这些偏好是个人在每种语言的编程时间、调试效果、语法、执行效率以及各种技术和文化的原因导致的结果。为了解决这些问题，我们将 ROS 设计成了语言中立性的框架结构。ROS 现在支持许多种不同的语言，例如 C++、Python、Octave 和 LISP，也包含其他语言的多种接口实现。

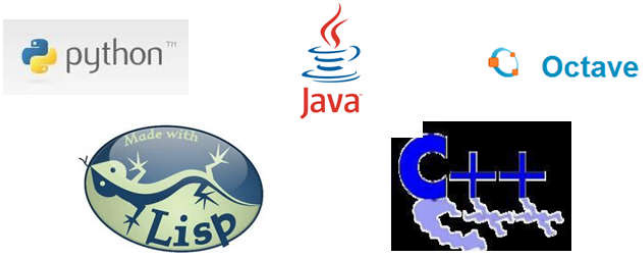

图 3.3 多语言支持

ROS 的特殊性主要体现在消息通讯层，而不是更深的层次。端对端的连接和配置利用 XML-RPC 机制进行实现，XML-RPC 也包含了大多数主要语言的合理实现描述。我们希望

ROS 能够利用各种语言实现的更加自然，更符合各种语言的语法约定，而不是基于 C 语言给各种其他语言提供实现接口。然而，在某些情况下利用已经存在的库封装后支持更多新的语言是很方便的，比如 Octave 的客户端就是通过 C++的封装库进行实现的。

为了支持交叉语言，ROS 利用了简单的、语言无关的接口定义语言去描述模块之间的消息传送。接口定义语言使用了简短的文本去描述每条消息的结构，也允许消息的合成，例如下图就是利用接口定义语言描述的一个点的消息：

```
Header header
Point32[] pts
ChannelFloat32[] chan
```

图 3.4 消息描述

每种语言的代码产生器就会产生类似本种语言目标文件，在消息传递和接收的过程中通过 ROS 自动连续并行的实现。这就节省了重要的编程时间，也避免了错误：之前 3 行的接口定义文件自动的扩展成 137 行的 C++代码，96 行的 Python 代码，81 行的 Lisp 代码和 99 行的 Octave 代码。因为消息是从各种简单的文本文件中自动生成的，所以很容易列举出新的消息类型。在编写的时候，已知的基于 ROS 的代码库包含超过四百种消息类型，这些消息从传感器传送数据，使得物体检测到了周围的环境。

最后的结果就是一种语言无关的消息处理，让多种语言可以自由的混合和匹配使用。

（3）精简与集成

大多数已经存在的机器人软件工程都包含了可以在工程外重复使用的驱动和算法，不幸的是，由于多方面的原因，大部分代码的中间层都过于混乱，以至于很困难提取出它的功能，也很难把它们从原型中提取出来应用到其他方面。

为了应对这种趋势，我们鼓励将所有的驱动和算法逐渐发展成为和 ROS 没有依赖性单独的库。ROS 建立的系统具有模块化的特点，各模块中的代码可以单独编译，而且编译使用的 CMake 工具使它很容易的就实现精简的理念。ROS 基本将复杂的代码封装在库里，只是创建了一些小的应用程序为 ROS 显示库的功能，就允许了对简单的代码超越原型进行移植和重新使用。作为一种新加入的有优势，单元测试当代码在库中分散后也变得非常的容易，一个单独的测试程序可以测试库中很多的特点。

ROS 利用了很多现在已经存在的开源项目的代码，比如说从 Player 项目中借鉴了驱动、运动控制和仿真方面的代码，从 OpenCV 中借鉴了视觉算法方面的代码，从 OpenRAVE 借鉴了规划算法的内容，还有很多其他的项目。在每一个实例中，ROS 都用来显示多种多样的配置选项以及和各软件之间进行数据通信，也同时对它们进行微小的包装和改动。ROS 可以不断的从社区维护中进行升级，包括从其他的软件库、应用补丁中升级 ROS 的源代码。

（4）工具包丰富

为了管理复杂的 ROS 软件框架，我们利用了大量的小工具去编译和运行多种多样的 ROS 组建，从而设计成了内核，而不是构建一个庞大的开发和运行环境。

这些工具担任了各种各样的任务，例如，组织源代码的结构，获取和设置配置参数，

形象化端对端的拓扑连接，测量频带使用宽度，生动的描绘信息数据，自动生成文档等等。尽管我们已经测试通过像全局时钟和控制器模块的记录器的核心服务，但是我们还是希望能把所有的代码模块化。我们相信在效率上的损失远远是稳定性和管理的复杂性上无法弥补的。

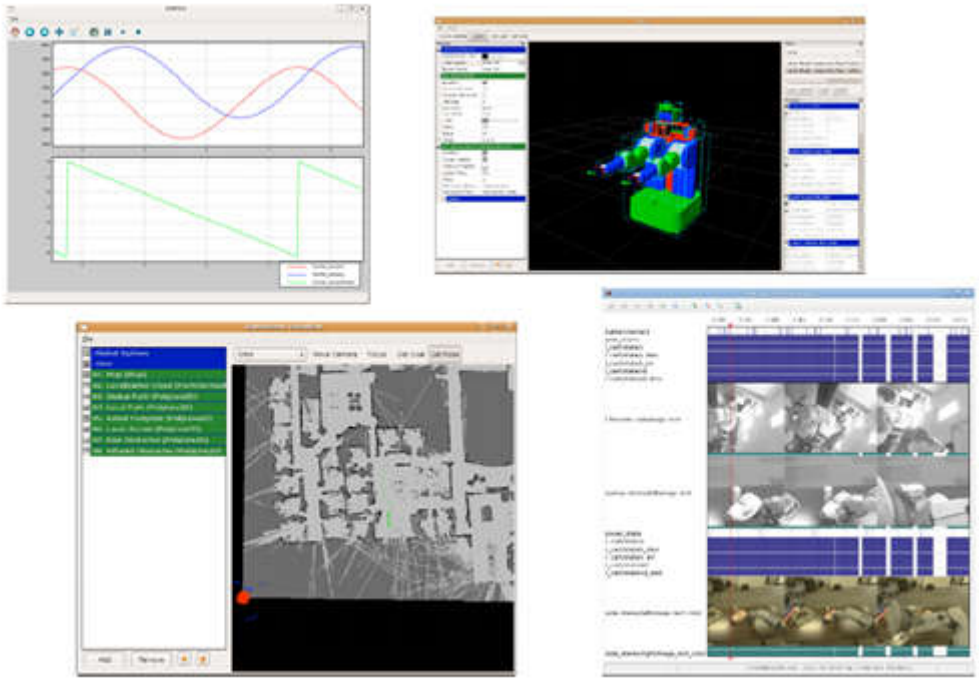

图 3.5 运行示例

（5）免费且开源

ROS 所有的源代码都是公开发布的。我们相信这将必定促进 ROS 软件各层次的调试，不断的改正错误。虽然像 Microsoft Robotics Studio 和 Webots 这样的非开源软件也有很多值得赞美的属性，但是我们认为一个开源的平台也是无可为替代的。当硬件和各层次的软件同时设计和调试的时候这一点是尤其真实的。

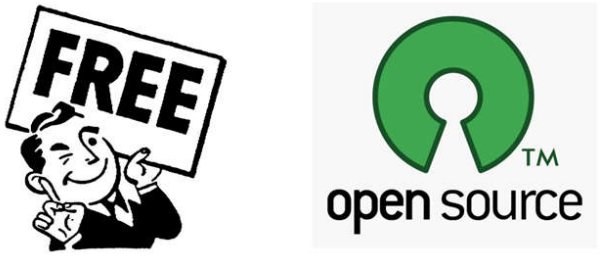

图 3.6 ros 特点

ROS 以分布式的关系遵循这 BSD 许可，也就是说允许各种商业和非商业的工程进行开发。ROS 通过内部处理的通讯系统进行数据的传递，不要求各模块在同样的可执行功能上连接在一起。如此，利用 ROS 构建的系统可以很好的使用他们丰富的组件：个别的模块可以包含被各种协议保护的软件，这些协议从 GPL 到 BSD，但是许可的一些“污染物”

将在模块的分解上就完全消灭掉。

3.3 ROS 总体框架

根据 ROS 系统代码的维护者和分布来标示，主要有两大部分：

（1）main：核心部分，主要由 Willow Garage 公司和一些开发者设计、提供以及维护。它提供了一些分布式计算的基本工具，以及整个 ROS 的核心部分的程序编写。

（2）universe：全球范围的代码，有不同国家的 ROS 社区组织开发和维护。一种是库的代码，如 OpenCV、PCL 等；库的上一层是从功能角度提供的代码，如人脸识别，他们调用下层的库；最上层的代码是应用级的代码，让机器人完成某一确定的功能。

一般是从另一个角度对 ROS 分级的，主要分为三个级别：计算图级、文件系统级、社区级。

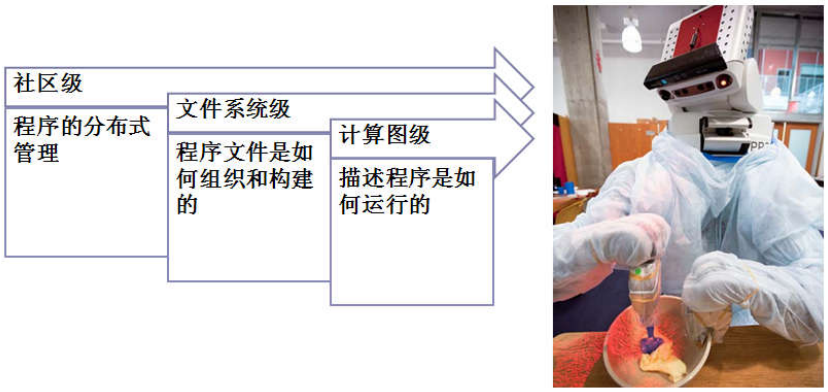

图 3.7 总体框架

3.3.1 计算图级

计算图是 ROS 处理数据的一种点对点的网络形式。程序运行时，所有进程以及他们所进行的数据处理，将会通过一种点对点的网络形式表现出来。这一级主要包括几个重要概念：节点（node）、消息（message）、主题（topic）、服务（service）。

图 3.8 计算图级

（1）节点

节点就是一些执行运算任务的进程。ROS 利用规模可增长的方式是代码模块化：一个系统就是典型的由很多节点组成的。在这里，节点也可以被称之为“软件模块”。我们使

用“节点”使得基于 ROS 的系统在运行的时候更加形象化：当许多节点同时运行时，可以很方便的将端对端的通讯绘制成一个图表，在这个图表中，进程就是图中的节点，而端对端的连接关系就是其中弧线连接。

（2）消息

节点之间是通过传送消息进行通讯的。每一个消息都是一个严格的数据结构。原来标准的数据类型（整型，浮点型，布尔型等等）都是支持的，同时也支持原始数组类型。消息可以包含任意的嵌套结构和数组（很类似于 C 语言的结构 structs）。

（3）主题

消息以一种发布/订阅的方式传递。一个节点可以在一个给定的主题中发布消息。一个节点针对某个主题关注与订阅特定类型的数据。可能同时有多个节点发布或者订阅同一个主题的消息。总体上，发布者和订阅者不了解彼此的存在。

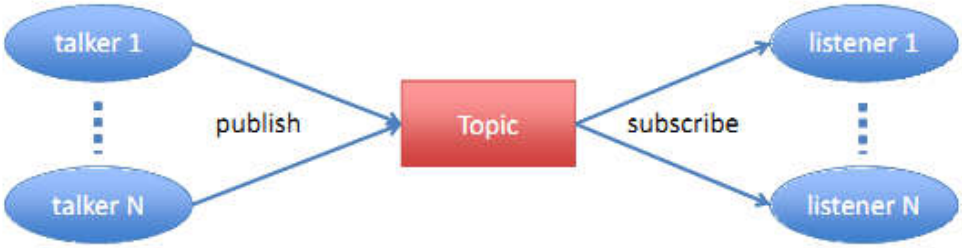

图 3.9 主题

（4）服务

虽然基于话题的发布/订阅模型是很灵活的通讯模式，但是它广播式的路径规划对于可以简化节点设计的同步传输模式并不适合。在 ROS 中，我们称之为一个服务，用一个字符串和一对严格规范的消息定义：一个用于请求，一个用于回应。这类似于 web 服务器，web 服务器是由 URIs 定义的，同时带有完整定义类型的请求和回复文档。需要注意的是，不像话题，只有一个节点可以以任意独有的名字广播一个服务：只有一个服务可以称之为“分类象征”，比如说，任意一个给出的 URI 地址只能有一个 web 服务器。

在上面概念的基础上，需要有一个控制器可以使所有节点有条不紊的执行，这就是一个 ROS 的控制器（ROS Master）。

ROS Master 通过 RPC（Remote Procedure Call Protocol，远程过程调用）提供了登记列表和对其他计算图表的查找。没有控制器，节点将无法找到其他节点，交换消息或调用服务。

比如控制节点订阅和发布消息的模型如下：

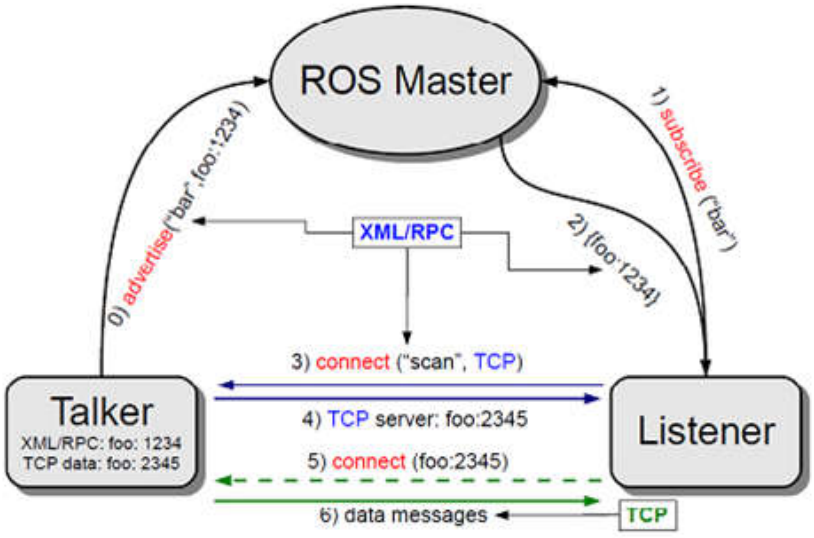

图 3.10 服务模型

ROS 的控制器给 ROS 的节点存储了主题和服务的注册信息。节点与控制器通信从而报告它们的注册信息。当这些节点与控制器通信的时候，它们可以接收关于其他以注册及节点的信息并且建立与其它以注册节点之间的联系。当这些注册信息改变时控制器也会回馈这些节点，同时允许节点动态创建与新节点之间的连接。

节点与节点之间的连接是直接的，控制器仅仅提供了查询信息，就像一个 DNS 服务器。节点订阅一个主题将会要求建立一个与出版该主题的节点的连接，并且将会在同意连接协议的基础上建立该连接。

3.3.2 文件系统级

ROS 文件系统级指的是在硬盘上面查看的 ROS 源代码的组织形式。

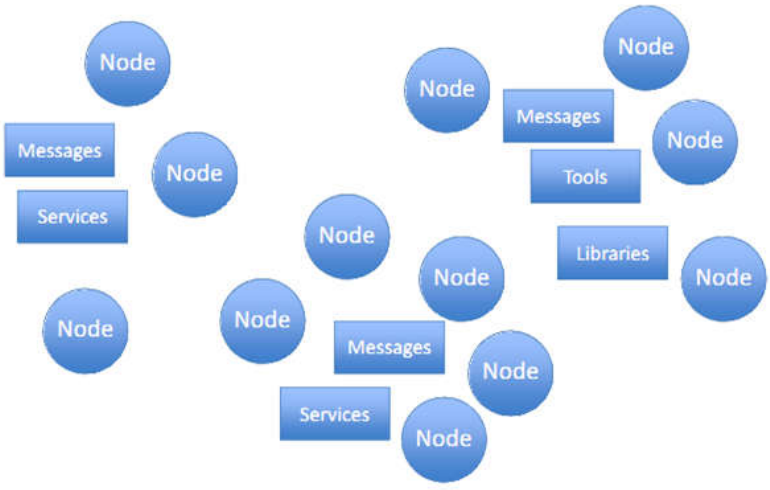

图 3.11 文件系统级

ROS 中有无数的节点、消息、服务、工具和库文件，需要有效的结构去管理这些代码。在 ROS 的文件系统级，有以下几个重要概念：包（package）、堆（stack）等。

（1）包：

图 3.12 包

ROS 的软件以包的方式组织起来。包包含节点、ROS 依赖库、数据套、配置文件、第三方软件、或者任何其他逻辑构成。包的目标是提供一种易于使用的结构以便于软件的重复使用。总得来说，ROS 的包短小精干。

（2）堆

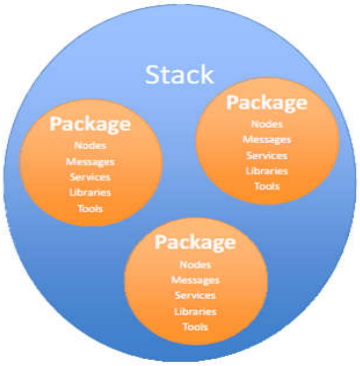

图 3.13 堆

堆是包的集合，它提供一个完整的功能，像“navigation stack”。Stack 与版本号关联，同时也是如何发行 ROS 软件方式的关键。

ROS 是一种分布式处理框架。这使可执行文件能被单独设计，并且在运行时松散耦合。这些过程可以封装到包（Packages）和堆（Stacks）中，以便于共享和分发。

2.3.3 社区级

ROS 的社区级概念是 ROS 网络上进行代码发布的一种表现形式。结构如下图所示：

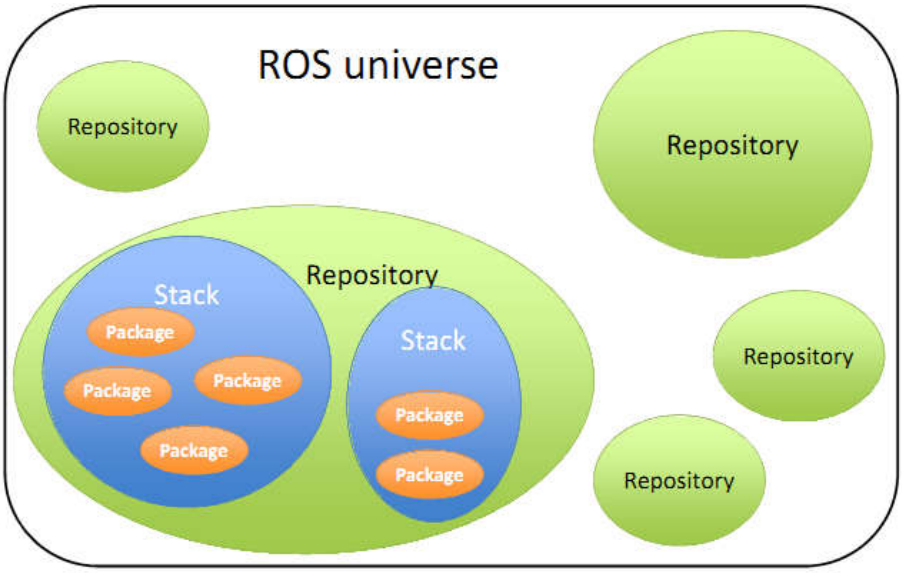

图 3.16 社区级

代码库的联合系统，使得协作亦能被分发。这种从文件系统级别到社区一级的设计让独立地发展和实施工作成为可能。正是因为这种分布式的结构，使得 ROS 迅速发展，软件仓库中包的数量指数级增加。

第 4 章　目标跟踪技术

4.1 视觉目标跟踪概述

4.1.1 视觉跟踪介绍

这里说的目标跟踪，是通用单目标跟踪，第一帧给个矩形框，这个框在数据库里面是人工标注的，在实际情况下大多是检测算法的结果，然后需要跟踪算法在后续帧紧跟住这个框。

通常目标跟踪面临由几大难点：外观变形，光照变化，快速运动和运动模糊，背景相似干扰等，如下图：

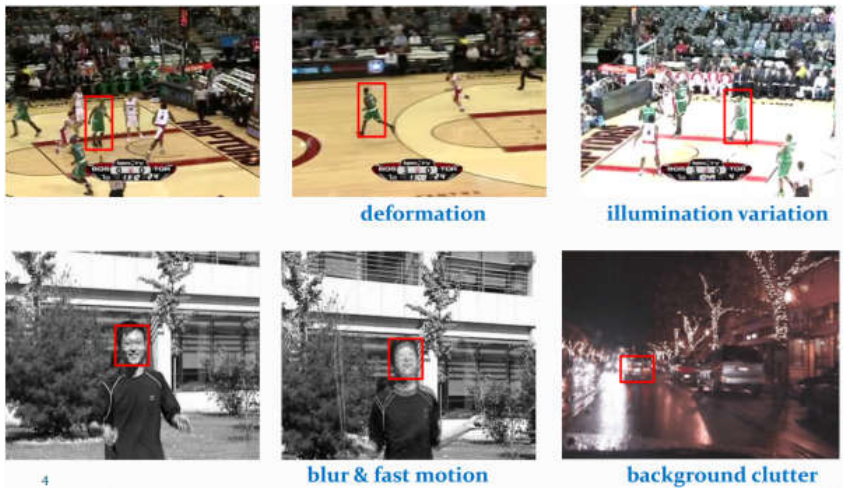

图 4.1 光照干扰

平面外旋转，平面内旋转，尺度变化，遮挡和出视野等情况，如下图：

图 4.2 运动干扰

正是因为这些情况才让目标跟踪变得很难。因此要建立一个鲁棒的跟踪系统，主要考虑的要求有：鲁棒性，适应性及实时性。

目标跟踪算法一般有以下几个流程：

（1）获得一种对跟踪目标的适当描述。

（2）包含特定的上下文信息。

（3）自适应的特征算子，包含有梯度特征，颜色特征，纹理特征及时-空特征等。

总的来说，大多视觉跟踪目标都会包含图像输入，外观特征描述，上下文信息融合及模型更新等几个步骤。不同的方法侧重点不同，采用的跟踪框架也会不同。

4.1.2 视觉目标跟踪分类

大家比较公认分为两大类：生成模型方法和判别模型方法，目前比较流行的是判别类方法，也叫检测跟踪(tracking-by-detection)，为保持完整性，以下简述。

（1）生成类方法

在当前帧对目标区域建模，下一帧寻找与模型最相似的区域就是预测位置，比较著名的有卡尔曼滤波，粒子滤波，mean-shift 等。举个例子，从当前帧知道了目标区域 80%是红色，20%是绿色，然后在下一帧，搜索算法就像无头苍蝇，到处去找最符合这个颜色比例的区域。

（2）判别类方法

此方法的经典套路图像特征+机器学习，当前帧以目标区域为正样本，背景区域为负样本，机器学习训练分类器，下一帧用训练好的分类器找最优区域。

判别类方法的最新发展就是相关滤波类方法，correlation filter 简称 CF，或 discriminative correlation filter 简称 DCF，和深度学习(Deep ConvNet based)类方法。

判别类与生成类方法最大的区别，是分类器训练过程中用到了背景信息，这样分类器专注区分前景和背景，判别类方法普遍都比生成类好。举个例子，在训练时告诉 tracker 目标 80%是红色，20%是绿色，还告诉它背景中有橘红色，要格外注意别搞错了，这样的分类器知道更多信息，效果也肯定更好。tracking-by-detection 和检测算法非常相似，如经典行人检测用 HOG+SVM。Struck 用到了 haar+structured output SVM，跟踪中为了尺度自适应也需要多尺度遍历搜索，区别仅在于跟踪算法对特征和在线机器学习的速度要求更高，检测范围和尺度更小而已。这点其实并不意外，大多数情况检测识别算法复杂度比较高不可能每帧都做，这时候用复杂度更低的跟踪算法就很合适了，只需要在跟踪失败或一定间隔以后再次检测去初始化 tracker 就可以了。经典判别类方法推荐 Struck 和 TLD，都能实时性能还行，Struck 是 2012 年之前最好的方法，TLD 是经典 long-term 类跟踪的代表，即使效果差一点但思想非常值得借鉴。

4.2 核相关的视觉目标跟踪算法（KCF）

4.2.1 算法描述

我们在这里是以“High-speed tracking with kernelized correlation filters”这篇论文来作为算法的基础。KCF 是一种判别式追踪方法，这类方法一般都是在追踪过程中训练一个目

标检测器，使用目标检测器去检测下一帧预测位置是否是目标，然后再使用新检测结果去更新训练集进而更新目标检测器。而在训练目标检测器时一般选取目标区域为正样本，目标的周围区域为负样本，当然越靠近目标的区域为正样本的可能性越大。

KCF 的主要特点：

（1）使用目标周围区域的循环矩阵采集正负样本，利用岭回归训练目标检测器，并成功的利用循环矩阵在傅里叶空间可对角化的性质将矩阵的运算转化为向量的 Hadamad 积，即元素的点乘，大大降低了运算量，提高了运算速度，使算法满足实时性要求。

（2）将线性空间的岭回归通过核函数映射到非线性空间，在非线性空间通过求解一个对偶问题和某些常见的约束，同样的可以使用循环矩阵傅里叶空间对角化简化计算。

（3）给出了一种将多通道数据融入该算法的途径。

4.2.2 线性回归

算法把跟踪问题抽象成一个线性回归模型的求解，设代表目标图像的输入为 z,权重为 w,输出为 $f(z)=w^Tz$， $w=[w_1,w_2,...,w_n]^T$ 需要通过训练得到， $z=[z_1,z_2,...,z_n]^T$ 是目标的特征。令样本最小化为 x_i，训练的目的是求最小化平方误差（MSE）下的 w：

$$\min_w \sum_i (f(x_i)-y_i)^2+\lambda \| w\|^2 \qquad (4\text{-}1)$$

其中 y_i 是期望回归值，λ 是防过拟合的正则化参数，用于控制系统的结构复杂性，保证分类器的泛化性能，和 SVM 中类似。其中 w 可以有以下形式：

$$w=(X^TX+\lambda I)^{-1}X^Ty \qquad (4\text{-}2)$$

其中矩阵 X 的每行为 x_i，即 $X=[x_1,x_2,...,x_n]^T$，I 为单位矩阵， $y=[y_1,y_2,...,y_n]^T$。

因为后面图片要在傅里叶域变换，牵涉到复数矩阵，所以我们将结果都统一写成复数形式：

$$w=(X^HX+\lambda I)^{-1}X^Hy \qquad (4\text{-}3)$$

其中*表示共轭，H 表示转置，即 $X^H=(X^*)^T$。当 X 由实数组成时，式（4-2）与（4-3）等价。

方程（4-3）要求逆矩阵，在大规模矩阵运算中，复杂度很大，运算很费力，如果 X 有特殊的构造，那么可以克服矩阵求逆带来的困扰，下面就带来怎样高效的避开求逆运算。

4.2.3 循环偏移

一个样本 x 可以表示为 n*1 的向量，如 $X=[x_1,x_2,...,x_n]^T$，我们把它作为基样本（base sample）,x_i 代表第 i 个样本的特征。我们的目标是训练一个分类器，包括基样本（正样本），

和通过移位得到的虚样本（负样本）。

假设有一个置换矩阵（permutation matrix）P,它能使基样本产生偏移。

$$P=\begin{bmatrix}0 & 0 & \dots & 0 & 1\\ 1 & 0 & \dots & 0 & 0\\ 0 & 1 & \dots & 0 & 0\\ & & \dots & & \\ 0 & 0 & \dots & 1 & 0\end{bmatrix} \tag{4-4}$$

X 乘一次 P，X 中的元素就向右循环偏移一位，当然如果是负数，也可以向左偏移，由此得到负样本 $P_x=[x_n,x_1,x_2,\dots,x_{n-1}]^T$ 。

所以由一个向量 $x\in R^n$ 可以通过不断地乘上排列矩阵得到 n 个循环移位向量，将这 n 个向量依序排列到一个矩阵中，就形成了 x 生成的循环矩阵，表示为 $C(x)$ 。

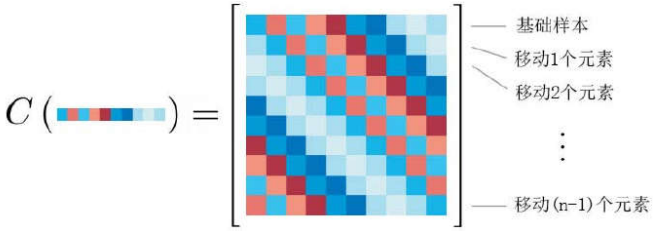

图 4.3 循环矩阵示意图

4.2.4 循环矩阵

前面 4.2.2 节末尾说过如果 X 有特殊的构造，那么就可以克服矩阵求逆带来的困扰，其实这里的特殊构造是指 X 每一行由基样本 x 的循环偏移向量组成，如下：

$$X=C(x)=\begin{bmatrix}x_1 & x_2 & x_3 & \dots & x_n\\ x_n & x_1 & x_2 & \dots & x_{n-1}\\ x_{n-1} & x_n & x_1 & \dots & x_{n-2}\\ & & \dots & & \\ x_2 & x_3 & x_4 & \dots & x_1\end{bmatrix} \tag{4-5}$$

如果用图像形象化就是：

图 4.4 垂直循环位移示例

中间的基样本图像分别左移和右移得到其余的样本，而且循环矩阵所具备的性质正是

帮助对式（4-3）求解的关键。

循环矩阵的作用：

（1）能够被离散傅里叶矩阵对角化，使得矩阵求逆转换为特征值求逆的性质；

（2）能够将式（4-3）转换到频域进行运算，应用离散傅里叶变换（DFT）提高运算速度。

其中 X 可以表示为：

$$X = Fdiag(\hat{x})F^H \tag{4-6}$$

其中 F 是离散傅里叶矩阵（DFT matrix），$\hat{x}$ 表示对 x 进行离散傅里叶变换，$diag$ 表示向量对角化。F 是常量，可以表示为：

$$F = \frac{1}{\sqrt{n}}\begin{bmatrix} 1 & 1 & \dots & 1 & 1 \\ 1 & w & \dots & w^{n-2} & w^{n-1} \\ 1 & w^2 & \dots & w^{2(n-2)} & w^{2(n-1)} \\ & & \dots & & \\ 1 & w^{n-1} & \dots & w^{(n-1)(n-2)} & w^{(n-1)^2} \end{bmatrix} \tag{4-7}$$

4.2.5 整合公式

这一节是特别重要的，通过推导得出一个简单地公式，利用这个公式可以轻松的计算出 w。我们将式（4-6）代入到式（4-3）中（其中 X^H 也是循环矩阵），结果就出来了。以下是推导过程：

$$\begin{aligned} w &= (Fdiag(\hat{x}^*)F^H Fdiag(\hat{x})F^H + \lambda F^H)^{-1} Fdiag(\hat{x}^*)F^H y \\ &= (Fdiag(\hat{x}^* \otimes \hat{x} + \lambda)F^H)^{-1} Fdiag(\hat{x}^*)F^H y \\ &= Fdiag(\frac{\hat{x}^*}{\hat{x}^* \otimes \hat{x} + \lambda})F^H y \end{aligned} \tag{4-8}$$

注意：这里的分号是点除运算，就是对应元素相除。

利用反对角性质：$Fdiag(y)F^H = C(F^{-1}(y))$ 得到：

$$w = C(F^{-1}(\frac{\hat{x}^*}{\hat{x}^* \otimes \hat{x} + \lambda}))y \tag{4-9}$$

再利用循环矩阵卷积性质：$F(C(x)y) = F^*(x) \otimes F(y)$ 得到：

$$F(w)=F(C(x)y)=(\frac{\hat{x}^{*}}{\hat{x}^{*}\otimes\hat{x}+\lambda})^{*}\otimes F(y)=\frac{\hat{x}^{*}\otimes\hat{y}}{\hat{x}^{*}\otimes\hat{x}+\lambda} \quad (4\text{-}10)$$

这样就可以使用向量的点积运算取代矩阵运算，大大提高了计算速度。

4.2.6 核技巧（Kernel trick）简要概述

上面所做的都是线性回归，但在现实种很多问题的解都是非线性的。非线性问题往往不好求解，所以希望能用解线性分类的方法解决这个问题。所采用的方法是进行一个非线性变换，将非线性问题变换为线性问题，通过求解变化后的线性问题的方法求解原来的非线性问题。这里采用的非线性变换就是采用核技巧。

前面几节中的回归问题就是解 $f(z)=w^{T}z$ 中的 w,而式（4-10）给出里解，但 w 是 x 的线性表示，当运用到实际非线性问题中误差会很大，所以现在需要用 x 的非线性组合表示 w：

$$w=\sum_{i}\alpha_{i}\varphi(x_{i}) \quad (4\text{-}11)$$

假设存在 $\varphi(x)\bullet\varphi(x')=\kappa(x,x')$，等号左边为点积，等号右边 $\kappa(x,x')$ 称为核函数。所以，原线性回归问题就可以表示为：

$$f(z)=(\sum_{i}\alpha_{i}\varphi(x_{i}))^{T}\bullet z=\sum_{i}\alpha_{i}\kappa(x_{i},z) \quad (4\text{-}12)$$

4.2.7 快速核回归

由上一节可知，最终的线性回归变换为了式（4-12）的非线性回归问题，现在要做的就是求出 α，α 为元素 α_{i} 的向量。

我们可以得到核函数下的非线性回归问题的解：

$$\alpha=(K+\lambda I)^{-1}y \quad (4\text{-}13)$$

其中 K 是 n*n 矩阵，$K_{ij}=\kappa(x_{i},x_{j})$。那么又回到了求解方程（4-3）的方法，即如果 K 是循环矩阵，计算量将大大降低，解决这个问题还得看核函数 $\kappa(x_{i},x_{j})$。

定理 1：存在任意偏移矩阵 M，都有 $\kappa(x,x')=\kappa(Mx,Mx')$，则 K 为循环矩阵，其中 $K_{ij}=\kappa(x_{i},x_{j})$。

满足以上定理的核函数有：高斯核，线性核，多项式核核及一些加性核函数。所以可将（4-13）对角化，得到：

$$\hat{\alpha} = \frac{\hat{y}}{\hat{k}^{xx} + \lambda} \tag{4-14}$$

k^{xx}是核矩阵$K = C(k^{xx})$的第一行，$\wedge$表示向量的 DFT 变换。

k^{xx}对于定义一个更通用的核相关很有用。任意俩个向量x和x'的核相关，即向量$k^{xx'}$的计算如下：

$$\hat{k}^{xx'} = \kappa(x', P^{i-1}x) \tag{4-15}$$

它包含了俩个参数的不同相对位移所得到的核相关，那么$\hat{k}^{xx}$则是 x 和它自身在频域的核相关，我们称作“核自相关”，类似线性情况。

这种类似可以进一步推广，由于一个核等价于高维空间中的点积$\varphi(\bullet)$，式（4-15）的另一种形式是：

$$k_i^{xx'} = \varphi^T(x')\varphi(P^{i-1}x) \tag{4-16}$$

这是x和x'在高维空间$\varphi(\bullet)$的互相关。

注意到我们只需要在核自相关上进行计算和操作，这是一个 n*1 的向量，随样本数量线性增加。这与传统核方法正好相反，传统方法需要计算一个 n*n 的核矩阵，用样本进行了二次缩放。由于我们知道 K 的确切结构，可以做的比通用算法更好。

由于 tracking-by-detection 方法中使用了很多转换后的图像块，因此对α取最优不是唯一可以加速的方法，下面我们将介绍循环移位模型在检测阶段以及核相关的计算方面的作用。

4.2.8 快速检测目标

通过上面的方法可以训练得到α，相当于得到了线性回归方程中的w，所以现在可以进行目标检测。

通过构建测试样本和训练样本的核函数矩阵如下：

$$K^z = C(k^{xz}) \tag{4-17}$$

其中k^{xz}是这个循环矩阵的第一行组成的向量，这样就可以同时计算基于测试样本 z 的循环偏移构成的所有测试样本的响应，即：

$$f(z) = (K^z)^T \alpha \tag{4-18}$$

注意，这里的$f(z)$不同于式（4-12），它是一个向量，由基于基样本 z 不同循环偏移下的响应值组成。根据循环矩阵的性质，上式变换到 DFT 域之后，即：

$$\hat{f}(z)=\hat{k}^{\hat{xz}}\otimes\hat{\alpha} \tag{4-19}$$

由此看来，在所有位置估计$f(z)$的值可以看做对核k^{xz}进行空间域的滤波，每个$f(z)$是k^{xz}中相邻核值按照学习到的参数α进行的线性组合。

4.2.9 快速计算核函数相关性

首先列出$k^{xx'}$的计算公式如下：

$$k^{xx'}=g(C(x)x') \tag{4-20}$$

其中，$g(x)$是核函数，$C(x)$是基于 x 为第一行的循环矩阵，根据循环矩阵的性质，可得：

$$K^{xx'}=g(F^{-1}(\hat{x}^{*}\otimes\hat{x}')) \tag{4-21}$$

其中F^{-1}表示傅里叶变换。

所以，对于多项式核，其计算公式如下：

$$K^{xx'}=g(F^{-1}(\hat{x}^{*}\otimes\hat{x}')+a)^{b} \tag{4-22}$$

对于高斯核，其计算公式如下：

$$K^{xx'}=\exp(-\frac{1}{\sigma^{2}}(\|x\|^{2}+\|x'\|^{2}-2F^{-1}(\hat{x_{c}}^{*}\otimes\hat{x_{c}}'))) \tag{4-23}$$

可以得到，我们能在$O(n\log n)$的复杂度下计算全部的核相关。

4.2.10 多通道数据处理

论文中在提取目标区域的特征时可以是灰度特征，但是使用 HOG 特征能够取得更好的效果，那么 Hog 特征该如何加入前面提到的模型呢？

HOG 特征是将图像换分成较小的局部块，称为 cell，在 cell 里提取梯度信息，绘制梯度方向直方图，然后为了减小光照影响，将几个 cell 的方向直方图串在一起进行 block 归一化，最终将所有的 cell 直方图串联起来就是图像的特征。

那么按照传统的方式一张图像就提取出一个向量，但是这个向量怎么用啊？我们又不能通过该向量的移位来获得采样样本，因为，你想啊，把直方图的一个 bin 循环移位有什么意义啊？

所以论文中 HOG 特征的提取是将 sample 区域划分成若干的区域，然后再每个区域提取特征，代码中是在每个区域提取了 32 维特征，即$3\times nOrients+5$，其中$nOrients=9$就是梯度方向划分的 bin 个数，每个方向提取了 3 个特征，2 个是对方向 bin 敏感的，1 个是不

敏感的，另外 4 个特征是关于表观纹理的特征，还有一个是零，表示阶段特征。提取了 31 个特征(最后一个 0 不考虑)之后，不是串联起来，而是将每个 cell 的特征并起来，那么一幅图像得到的结果就是一个立体块，假设划分 cell 的结果是 $m \times n$,那么 fhog 提取结果就是 $m \times n \times 31$,我们称为 31 个方向通道。那么就可以通过 cell 的位移来获得样本，这样对应的就是每一通道对应位置的移位，所有样本的第 i 通道都是由生成图像的第 i 通道移位获得的，所以分开在每一个通道上计算，就可以利用循环矩阵的性质了。

然后，依据高斯核，扩展到多通道就是下式：

$$K^{xx'} = \exp(-\frac{1}{\sigma^2}(\| x \|^2 + \| x' \|^2 - 2F^{-1}(\sum_c \hat{x}_c^{*} \otimes \hat{x}_c'))) \tag{4-24}$$

4.3 KCF 算法实现

4.3.1 视觉目标检测框架

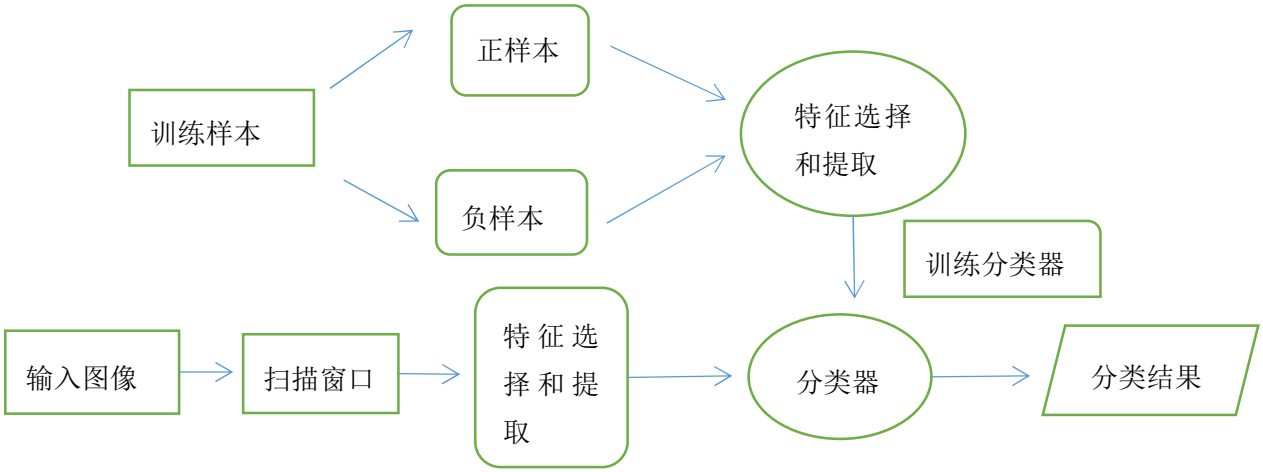

图 4.5 目标检测框架

目标检测分为以下几个步骤：

（1）训练样本的创建

训练样本包括正样本和负样本，其中正例样本是指待检目标样本(例如人脸或汽车等)，负样本指其它不包含目标的任意图片（如背景等），所有的样本图片都被归一化为同样的尺寸大小(例如，20x20)。

（2）特征提取

由图像或波形所获得的数据量是相当大的。例如，一个文字图像可以有几千个数据，一个心电图波形也可能有几千个数据。为了有效地实现分类识别，就要对原始数据进行变换，得到最能反映分类本质的特征。这就是特征选择和提取的过程。一般我们把原始数据组成的空间叫测量空间，把分类识别赖以进行的空间叫做特征空间，通过变换，可把在维数较高的测量空间中表示的模式变为在维数较低的特征空间中表示的模式。

（3）训练分类器

这得先明白分类器是什么？百度百科的解释是：“使待分对象被划归某一类而使用的分类装置或数学模型。”我觉得可以怎么理解，举个例子：人脑本身也算一个分类器（只是它强大到超乎想象而已），人对事物的识别本身也是一个分类的过程。人在成长或者学习过程中，会通过观察 A 类事物的多个具体事例来得到对 A 类事物性质和特点的认识，然后以后遇到一个新的物体时，人脑会根据这个事物的特征是否符合 A 类事物性质和特点，而将其分类为 A 类或者非 A 类。（这里只是用简单的二分类问题来说明）。那么训练分类器可以理解为分类器（大脑）通过对正样本和负样本的观察（学习），使其具有对该目标的检测能力（未来遇到该目标能认出来）。

从数学来表达，分类器就是一个函数 y=f(x)，x 是某个事物的特征，y 是类别，通俗的说就是例如，你输入张三的特征 x1，分类器就给你认出来这个是张三 y1，你输入李四的特征 x2，它就给你认出来这个是李四 y2。那么分类器是个函数，它的数学模型是什么呢？一次函数 y=kx+b？高次函数？等等好复杂的都有，我们需要先确定它的模型；确定了模型后，模型是不是由很多参数呢？例如上面的一次函数 y=kx+b 的 k 和 b，高斯函数的均值和方差等等。这个就可以通过什么最小化分类误差、最小化惩罚啊等等方法来确定，其实训练分类器好像就是找这些参数，使得达到最好的分类效果。

另外，为了使分类检测准确率较好，训练样本一般都是成千上万的，然后每个样本又提取出了很多个特征，这样就产生了很多的训练数据，所以训练的过程一般都是很耗时间的。

（1）利用训练好的分类器进行目标检测

得到了分类器就可以用来对你输入的图像进行分类了，也就是在图像中检测是否存在你想要检测的目标。一般的检测过程是这样的：用一个扫描子窗口在待检测的图像中不断的移位滑动，子窗口每到一个位置，就会计算出该区域的特征，然后用我们训练好的分类器对该特征进行筛选，判定该区域是否为目标。然后因为目标在图像的大小可能和你训练分类器时使用的样本图片大小不一样，所以就需要对这个扫描的子窗口变大或者变小（或者将图像变小），再在图像中滑动，再匹配一遍。

（2）学习和改进分类器

现在如果样本数较多，特征选取和分类器算法都比较好的情况下，分类器的检测准确度都挺高的了。但也会有误检的时候。所以更高级点的话就是加入了学习或者自适应，也就是说你把这张图分类错误了，我就把这张图拿出来，标上其正确的类别，再放到样本库中去训练分类器，让分类器更新、醒悟，下次别再给我弄错了。你怎么知道他弄错了？我理解是：大部分都是靠先验知识（例如目标本身存在着结构啊或者什么的约束）或者和跟踪（目标一般不会运动得太快）等综合来判断的。

综合来看，上面这个框架的过程是适合很多领域的，例如目标跟踪，语音识别等，那么这整个过程的关键点在哪呢？

（1）特征选取：

感觉目前比较盛行的有：Haar 特征、LBP 特征、HOG 特征和 Shif 特征等；他们各有千秋，得视你要检测的目标情况而定，例如：

拳头：纹理特征明显：Haar、LBP；

手掌：轮廓特征明显：HOG 特征（行人检测一般用这个）；

（2）分类器算法：

感觉目标比较盛行的有：SVM 支持向量机、AdaBoost 算法等；其中检测行人的一般是 HOG+SVM，OpenCV 中检测人脸的一般是 Haar+AdaBoost，OpenCV 中检测拳头一般是 LBP+ AdaBoost。

4.3.2 软件设计

我们采用 C++程序设计语言来实现目标跟踪的功能，并且用到了 OpenCV 开源计算机视觉库和 ROS 开源机器人操作系统，在 Ubuntu 平台下进行软件设计。

下面我们将对参数配置，目标跟踪，模型更新等模块进行阐述和分析。

（1）参数配置

我们通过一个构造函数来进行默认的参数配置,函数接口如下：

KCFTracker::KCFTracker(bool hog, bool fixed_window, bool multiscale, bool lab)

函数中的参数解释：

hog:我们选取 HOG 特征作为模型的特征。

fixed_window: 固定选取窗口的大小

multiscale: 采用多尺度特征

lab: Lab 颜色空间

这些参数默认都是 true。

函数中的部分默认参数变量：

lambda ： 模型更新参数，默认值是 0.0001

padding ：跟踪框的填充倍数，默认值是 2.5，即处理的跟踪区域是在以目标为中心，长宽分别是目标大小的 2.5 倍的跟踪框。

output_sigma_factor：高斯回归目标相对于目标大小的空间带宽，默认值是 0.125。

interp_factor：适应率，即更新模型的记忆因子（学习速率），hog 默认是 0.012，lab 默认值是 0.005。

sigma：高斯核参数，hog 默认是 0.6，lab 默认值是 0.4。

cell_size: HOG 特征中 cell 的大小，默认值是 4。

（2）跟踪模块

我们在读入一帧图像后，利用 KCF 算法在下一帧中对目标位置进行预测，具体步骤如下：

①首先以上一帧（预测）的目标中心为中心点，提取一个长宽都为目标 2.5 倍的图像块。（并且和模板大小进行了比较，得到一个尺度因子，得到的结果是较大边长按模板大小赋值，较小边长按尺度因子缩放。）

②提取 HOG/Gray 特征，下面一节会详细介绍如何提取 HOG 特征

③对提取到的特征点进行 Hanning 窗口（余弦）处理。因为循环偏移采样得到的样本在靠近边界处会有一些明显的边界线，导致图像变得不平滑，因此需要乘以一个余弦窗来

降低边缘像素的权重。

以上步骤的函数接口是：

cv::Mat KCFTracker::getFeatures(const cv::Mat & image, bool inithann, float scale_adjust)

④对提取到的特征进行快速傅里叶变换，用到的函数接口是：

cv::Mat fftd(cv::Mat img, bool backwards)

⑤计算 x（当前）与 z（样本）的核相关矩阵，用到的函数接口是：

cv::Mat KCFTracker::gaussianCorrelation(cv::Mat x1, cv::Mat x2)

⑥计算响应值，用到的函数接口是：

real(fftd(complexMultiplication(_alphaf, fftd(k)), true)

⑦计算响应的极大值和极小值，并得到其所在位置，用到的函数接口是：

cv::minMaxLoc(res, &pv_min, &pv, &pi_min, &pi);

⑧得到当前帧的目标的中心预测值。

该过程的流程图如图 4.6 所示：

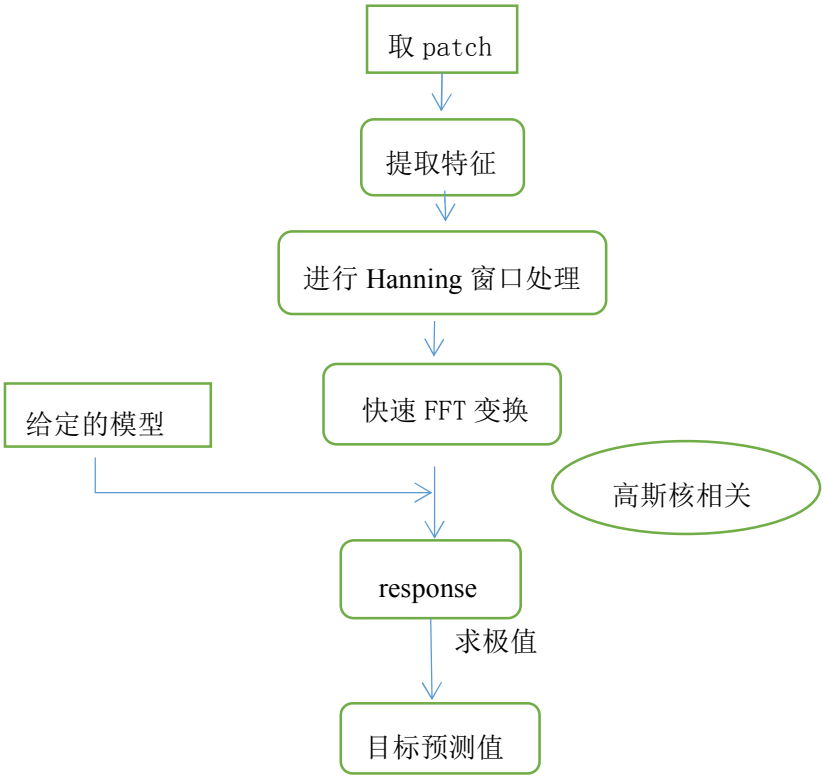

图 4.6 跟踪模块流程图

（3）模型更新模块

系统在目标检测过后进入模型更新的阶段，也就是我们所说的训练阶段，为下一帧检测做准备。

模型更新的主要步骤如下：

①首先以上一帧（预测）的目标中心为中心点，提取一个长宽都为目标 2.5 倍的图像块。

②提取 HOG/Gray 特征。

③对提取到的特征点进行 Hanning 窗口（余弦）处理。

④对提取到的特征进行快速傅里叶变换。

⑤用 x 对模型_tmpl 进行更新，按照如下方式：

```
_tmpl = (1 - train_interp_factor) * _tmpl + (train_interp_factor)*x
```

⑥计算 x 的高斯核自相关，按照如下方式：

```
cv::Mat k = gaussianCorrelation(x, x)
```

⑦计算 alphaf,按照如下方式：

```
cv::Mat alphaf = complexDivision(_prob, (fftd(k) + lambda));
```

⑧用 alpha 对模型_alphaf 进行更新，按照如下方式：

```
_alphaf=(1 - train_interp_factor) * _alphaf + (train_interp_factor)* alphaf;
```

该过程的流程图如图 4.7 所示：

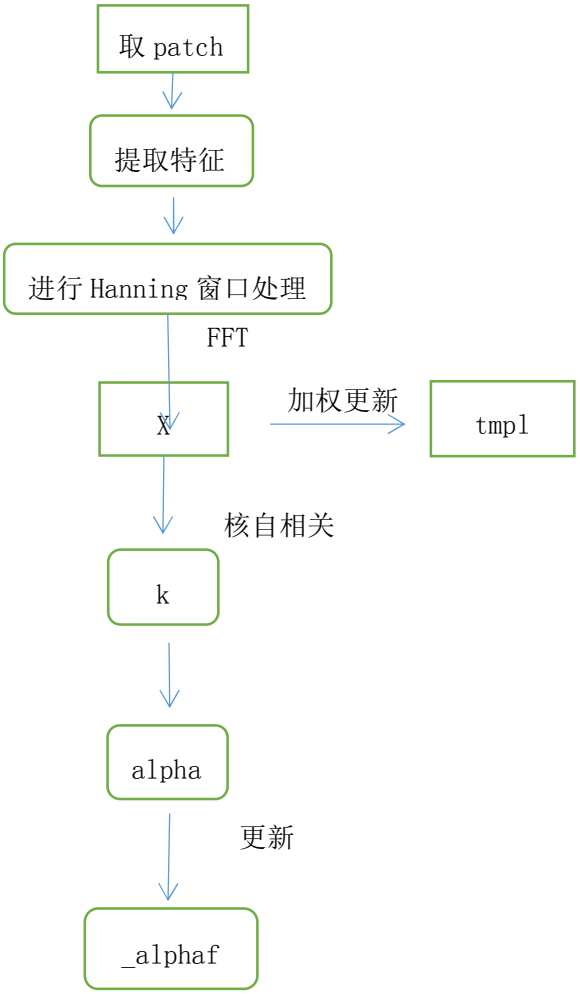

图 4.7 模型更新流程图

第 5 章 系统实现

5.1 实际场景运行

首先我们介绍一下启动步骤：

（1）启动 ros 环境

$roscore

（2）运行 ros 打开深度相机的节点

$roslaunch openni_launch openni.launch

（3）启动底盘程序

$roslaunch mrobot_bringup mrobot.launch

（4）启动追踪程序

$rosrun track_pkg kcf_node

（5）程序启动后，在图像窗口内用鼠标左键选取所要跟踪的目标

选定目标之后，小车会通过旋转将目标移动的相机的中心位置。目标在距离深度相机 1.5m 时开始跟踪，初始速度为 0.4m/s，速度随着距离的增大而增加。最大的识别距离是 5m。旋转速度初始为 0，最大的角速度为 0.75rad/s。

本系统由于 kinect 摄像头位置较低，所以我们将采用小腿绑目标的方式进行测试，如下图：

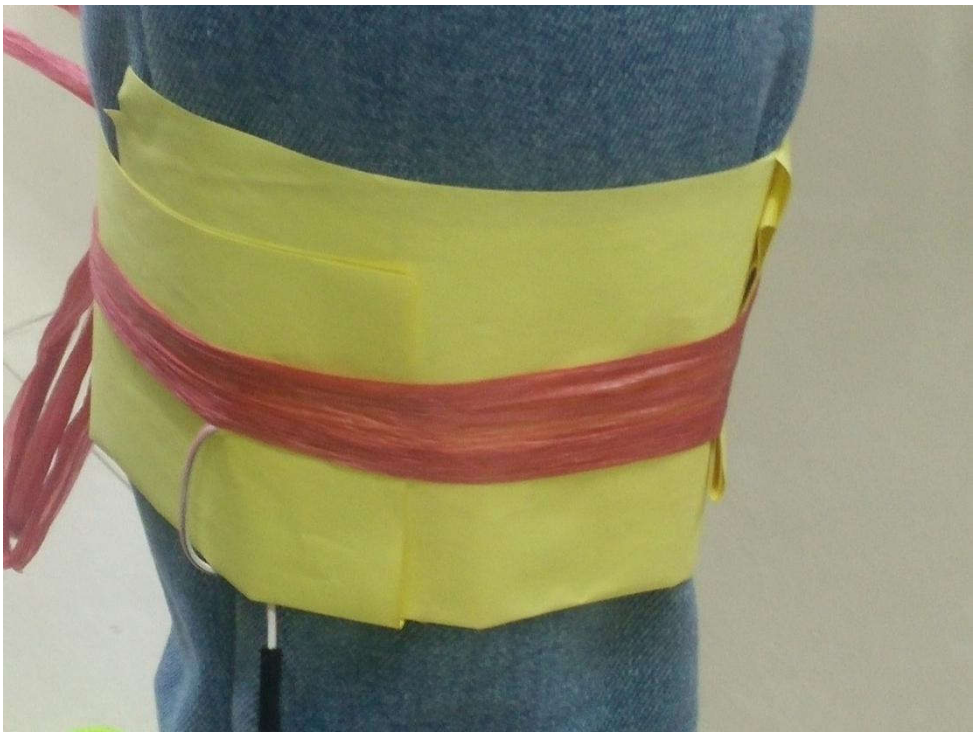

图 5.1 跟踪目标

目标的选取要尽量避免和背景由较多的重复，最好能有显著的特征，这样小车跟随的

时候会有不错的效果。

我们总共测试了四种情况，分别是：前进，后退，左转和右转。下图为测试时的场景：

图 5.2 测试场景

5.2 系统实验效果分析

在这里我们有个无法解决的缺点，就是在前进的时候，两边的轮子启动时间不一样，这就导致了在刚开始前进的时候小车的方向会偏离一点，不过在启动之后就会运行正常。

注意在测试的时候要保持小车的重心，否则小车会打滑。在前进时，小车会在目标距离大于 1.5m 时开始移动。在后退时，小车会在目标距离小于 1m 时开始后退，所以此时要小心不能碰到小车。左转和右转情况基本相同，在目标转过较大角度时，小车旋转的非常快，所以要把上面的东西固定好。

为了可以达到更好的效果，我们可以更换一对摩擦力较大的轮子，或者在比较粗糙的地面进行测试，但是不能有过陡的障碍，因为电机的扭矩较小，不能越过比较陡的障碍。

结 论

本系统在设计过程中，从最初的方案制定一直到选定方案，真的废了好大劲才最终选定用 KCF 的目标跟踪方法。在这个过程中，我查阅了大量的资料，从一开始的“懵懂”到最后的“成熟”，历尽了千难万险，我很庆幸自己没有放弃，始终相信自己能够完成。这个系统我觉得最重要的就是目标跟踪算法的实现，我最初选择的是基于蒙特卡罗方法的粒子滤波算法，但后来发现了跟踪效果更好地基于核相关的滤波算法，俩者相比，我认为后者更容易理解和实现。好在我找到了原作者写的论文，然后接下来就开始一步步去啃英文论文，在此期间瞬间感觉自己的英文水平更进了一步。总的来说，系统最终实现了，效果也是不错的。唯一的遗憾就是硬件和机械结构不太让我满意，导致我花费了比较的多时间去调试硬件。尽管我不是个完美的人，但我一直希望大学的最后一个作品能够尽量“完美”，虽然希望落空了吧，但重要的是真的在这个过程学到了好多的知识，让我对自己也有了一个重新的认知。

参考文献

1 刘志强.基于核相关滤波的高速目标跟踪算法研究与系统实现[D].西安电子科技大学硕士学位论文.2015.

2 赵璐璐.基于相关滤波的目标跟踪算法研究[D].北方工业大学硕士学位论文.2013.

3 罗雅愉.基于核相关滤波的目标跟踪算法研究[D].华南理工大学硕士学位论文.2015.

4 Henriques, João F, et al. "High-Speed Tracking with Kernelized Correlation Filters. " High Performance Computing, Networking, Storage and Analysis, SC14: International Conference for IEEE, 2014:375-386.

5 陈东成.基于机器学习的目标跟踪技术研究[D].中国科学院大学博士学位论文.2015.

6 贺超.基于 kinect 的移动机器人同时目标跟踪与壁障[D].太原理工大学硕士学位论文 2010.

7 朱志宇.粒子滤波算法及其应用[M].科学出版社.2010.

8 于仕琪，刘瑞祯.学习 OpenCV（中文版）[M].清华大学出版社.2009.

9 刘锦涛，张瑞雷.ROS 机器人程序设计（原书第 2 版）[M].机械工业出版 社.2016.

10 高翔，张涛.视觉 SLAM 十四讲-从理论到实践[M].电子工业出版社.2017.

11 Boddeti, Vishnu Naresh, T. Kanade, and B. V. K. V. Kumar. "Correlation Filters for Object Alignment." IEEE Conference on Computer Vision and Pattern ecognition IEEE Computer Society, 2013:2291-2298.

12 Kalal Z, Mikolajczyk K, Matas J. Tracking-Learning-Detection.[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2012, 34(7):1409.

致 谢

在我整个本科的学习生活中，我时刻能感受到指导老师和实验室同学对我的帮助和理解，以及舍友对我的容忍。在此，我想他们表达最真诚的感谢。

首先感谢我的指导导师岳洪伟老师，在此期间，岳老师对我的工程实践给予了充分的指导和肯定，使得我的综合能力能够快速提高。他的谆谆教诲和严谨的教学态度都使得我终身受益。

感谢实验室同学对我的学术指导和帮助，有好多次都是他们的答疑解惑才使得我能够搞懂某个知识点，帮助我加快了项目进度，使我能够如期完成项目设计。他们每个夜晚的陪伴也给了我很大的鼓舞，是自己感觉不是一个人在战斗。

感谢舍友对我的“不杀之恩”，以及他们对我的陪伴和鼓励。

特别感谢我的家人和父母，感谢他们对我的支持和理解，他们的关心是我前进路上的精神支柱，因为有他们的支持我才能度过人生道路上的重重困难，不断地前行。

再次由衷的感谢给予我帮助的老师，家人以及同学们！