大作业　Duckietown 智能小车协调控制

自 55 班　刘乐章　2015011471

2018 年 1 月 15 日

1　需求分析

目标　基于 Duckietown 小车平台，实现小车的远程控制与图像传输、处理，完成二维码小车跟随的任务。

综合考虑通信稳定性、软硬件兼容性与程序通用性等因素，选用 Ubuntu 16.04 平台，采用 ROS Kinetic 机器人操作系统实现。

2　总体设计

该程序的全部功能封装在 *duck_commander*ROS 包中。主要由节点和消息两部分构成。

2.1　节点 (node)

整个程序由三个节点组成 *duck_eye*, *duck_brain* 以及 *duck_keyboard* 。

• **duck_eye**　图像处理节点，Duck 的 “眼睛”。订阅上层节点图像信息，检测其中的二维码，并发送 RectLocation 消息将二维码的中心点横坐标 x 与面积 area 发布出去。

• **duck_brain**　运动控制节点，Duck 的 “大脑”。订阅 duck_eye 发布的话题，变换为电机控制的信息，并发送 Twisted2DStamp 消息将运动速度 v 与转弯角速度 omega 发布出去。

• **duck_keyboard**　键盘控制节点，Duck 的 “主人”。读取键盘输入信息，发送 Twisted2DStamp 消息将运动速度 v 与转弯角速度 omega 发布出去。

2.2　消息 (message)

除了 Duckietown 提供的消息之外，我们实现了两个自定义消息 *RectLoation* 和 $Twist2DStamped$。

• **RectLocation**　二维码位置消息。

• **Twist2DStamped**　小车运动消息。由头部 Header，速度 v 和转弯角速度 omega 组成。该消息继承自 Duckietown 内部消息，重新在 *duck_commander* 包中封装实现。

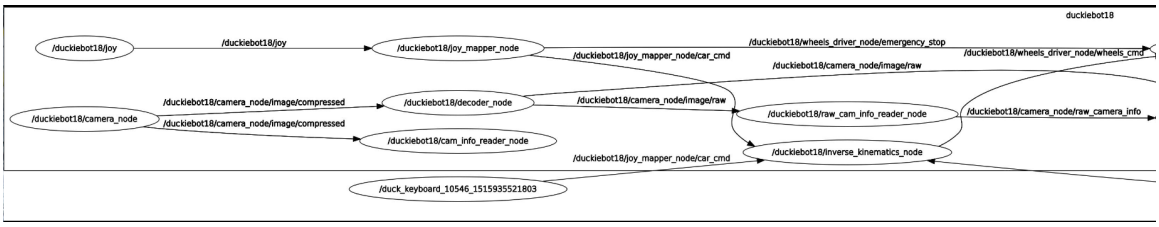

图 1: 程序节点图（左）

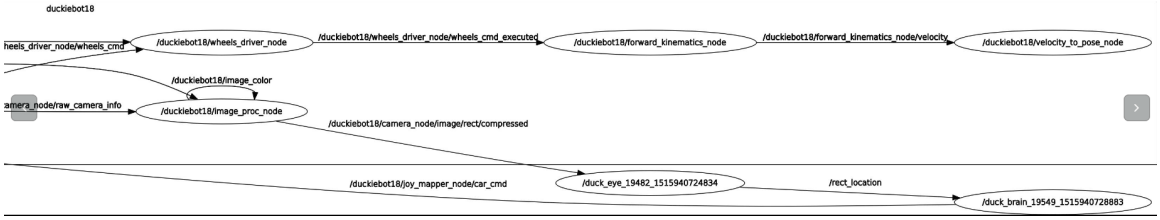

图 2: 程序节点图（右）

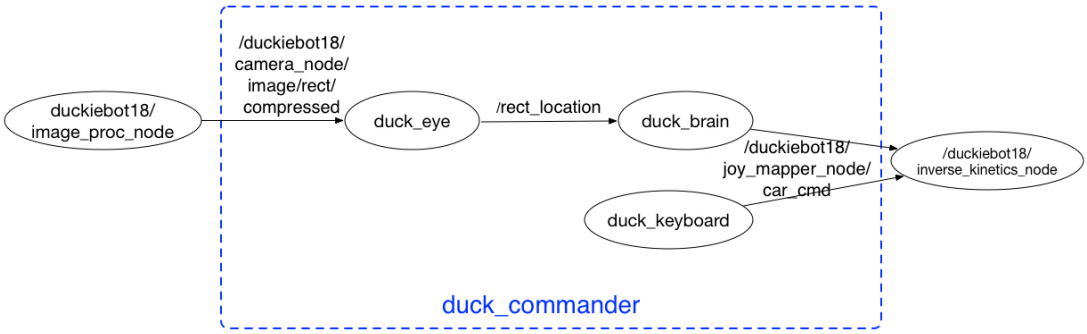

图 3: 个人实现的节点图

程序运行的节点图如图所示。

个人实现的 ROS 包 *duck_commander* 中的节点如图所示。

3 详细设计

3.1 图像处理节点 (duck_eye)

3.1.1 图像信息通信

图像信息通信由 ROS 话题订阅机制实现。相当与一个订阅者 (listener.py) 和一个发布者 (talker.py) 的结合。订阅者每收到一次消息，发布者便发布一个消息。

duck_eye 节点订阅 */duckiebot18/camera_node/image/rect/compressed* 消息，获取经过校正、压缩之后的图像。

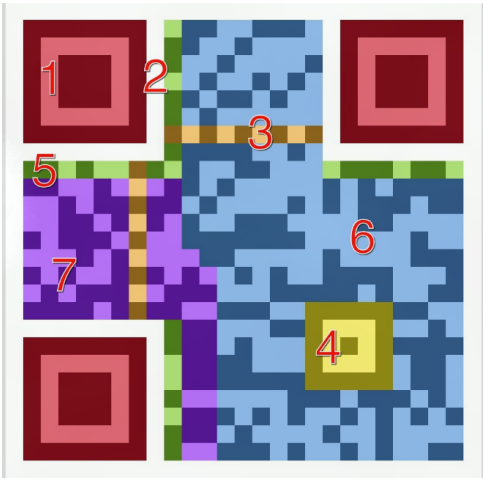

图 4: 二维码结构

获取输入图像后，该节点首先利用 cvBridge 进行类型转换，将其转换为 opencv 可以处理的图片类型。

接下来，该节点检测其中的二维码，并发送 RectLocation 消息将二维码的中心点横坐标 x 与面积 area 发布出去。

3.1.2 二维码检测

二维码检测主要利用到了二维码的形态信息。一个典型的二维码结构如图所示。

可以看到，其存在三个嵌套的正方形点，即“定位点”。我们的思路是：检测图片中的定位点，若发现了有三个定位点，即认为检测到了二维码。

首先，采用 Canny 算子检测边缘，结果如下图所示。

接下来对其进行形态学检测，找出五层嵌套的点，核心代码如图所示。

在找到所有定位点后，进行逻辑判断，若找到定位点数目等于 3, 则分别计算其中心做坐标。分别取 x，y 方向上最大值、最小值的平均值为二维码中心点坐标。取其 x，y 方向最大值最小值之差作为边长，并取边长乘积作为面积。将二维码的中心点横坐标 x 与面积 area 发布出去。

3.2 运动控制节点 (duck_brain)

3.2.1 运动信息通信

同样, 运动信息通信由 ROS 话题订阅机制实现。相当与一个订阅者 (listener.py) 和一个发布者 (talker.py) 的结合。订阅者每收到一次消息，发布者便发布一个消息。

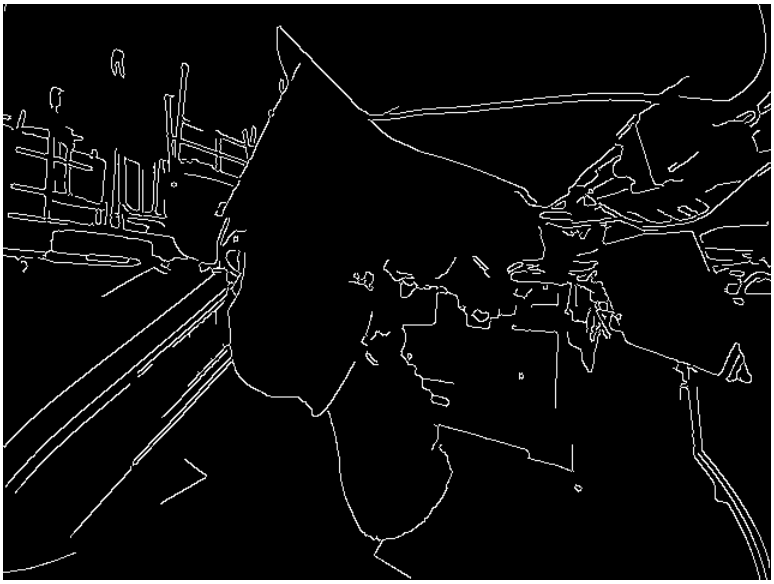

图 5: Canny 算子边缘检测

```python
def loc_barcode(img):
    # convert to grey
    grey = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # gaussian blur
    gb = cv2.GaussianBlur(grey, (5, 5), 0)

    # edge
    edge = cv2.Canny(gb, 100, 200)
    cv2.imshow("edge", edge)

    img_fc, contours, hierarchy = cv2.findContours(edge, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    hierarchy = hierarchy[0]
    found = []
    for i in range(len(contours)):
        k = i
        c = 0
        while hierarchy[k][2] != -1:
            k = hierarchy[k][2]
            c = c + 1
        if c >= 5:
            found.append(i)

    draw_img = img.copy()

    # find 3 mark points
    if len(found) == 3:
```

图 6: 形态学检测五层嵌套核心代码

```python
# global variables for PID and Exp Average
last_x = 320
last_area = 8000
alpha = 0.8


def car_control(x, area):
    # PD control for v
    e = area - 8000
    d = area - last_area
    Kp = -0.00005
    Kd = 0
    v = Kp * e + Kd * d

    # PD control for omega
    e = x - 320
    d = x - last_x
    Kp = 0.01
    Kd = 0.005
    omega = Kp * e + Kd * d
```

图 7: PD 控制代码实现

duck_brain 节点订阅 */rect_location* 消息，获取二维码中心横坐标 x 和面积 *area*。经过计算后，经 *Twist2DStamped* 消息发布速度 v，转弯角速度 omega 信息至 */duckiebot18/joy_mapper_node/car_cmd* 话题，以完成小车的运动控制。

3.2.2 PD 控制

对于小车的运动控制，我们采用 PD 控制的策略。

取小车速度 v，转弯角速度 *omega* 为控制量，二维码中心横坐标 x，面积 *area* 为输入量进行 PD 控制。由物理常识可得，以面积 *area* 控制小车速度 v，以中心坐标 x 控制转弯角速度 *omega*，其控制框图如图所示。

其中，选取平衡点面积 $area' = 8000$，平衡点横坐标 $x' = 320$。对于数字控制系统，以差分实现微分。调整参数，代码如图所示。

3.2.3 指数加权平滑

观察 *duck_eye* 节点二维码检测信息，可以看到存在明显的波动。为此，我们想到采用计算机网络课程中学到到指数加权平滑的方法来处理，其公式如下：

$$S_t = \alpha x_t + (1 - \alpha) S_{t-1}$$

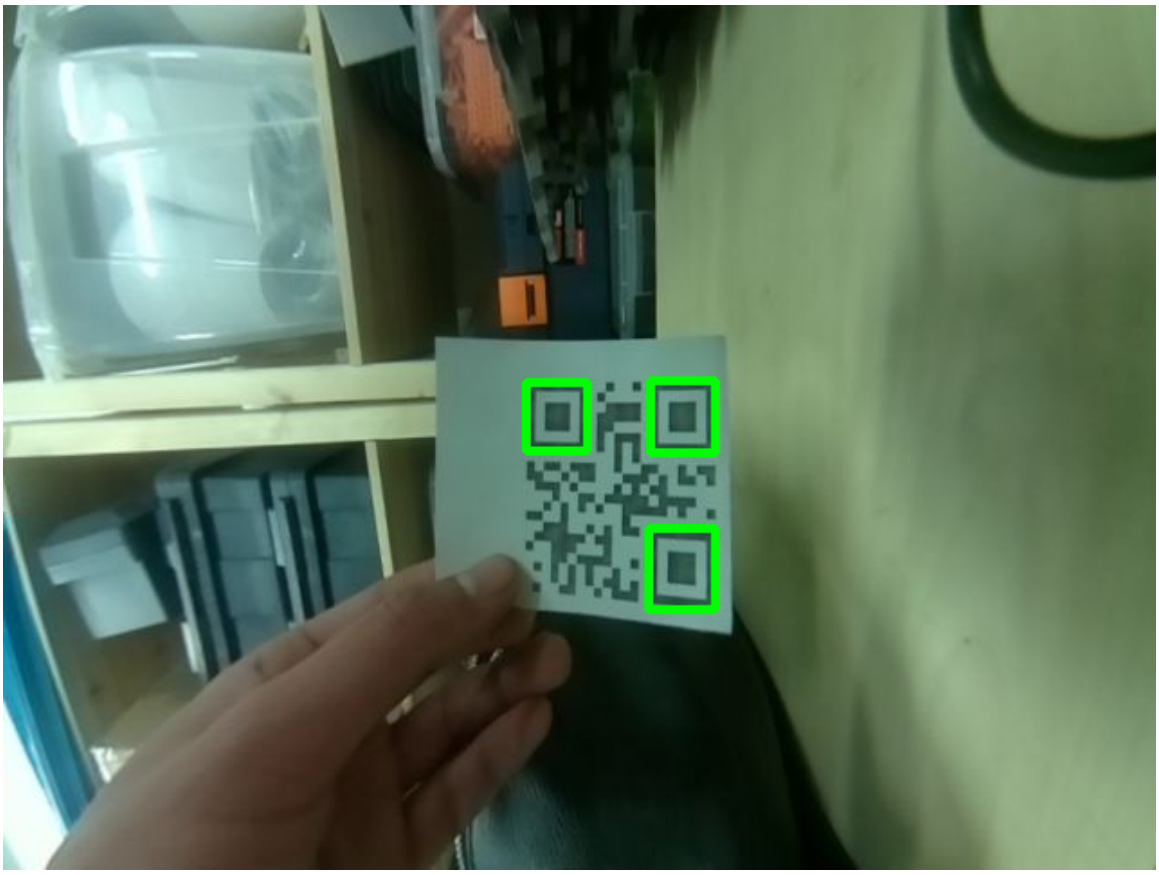

图 8: 二维码检测运行结果

$$S_0 = x_0$$

取 $\alpha = 0.8$，即可得到较好的平滑结果。

3.3 键盘控制节点 (duck_keyboard)

3.3.1 按键检测

采用键盘控制小车运动，其运动信息通信与运动控制节点大致相同。需要解决的是按键检测问题。

采用 py-getch 包提供的 getch 函数，可以实现按键检测的功能。

4 结果分析

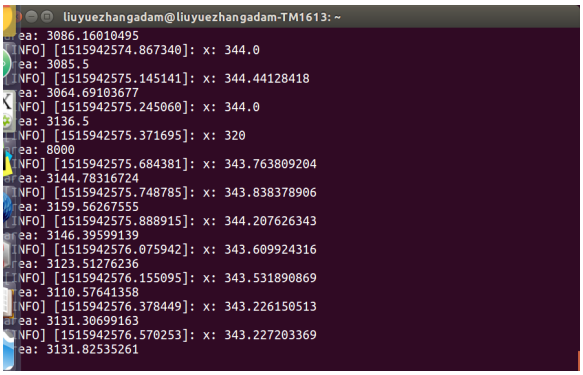

图 9: 图像处理节点 duck_ eye 运行结果

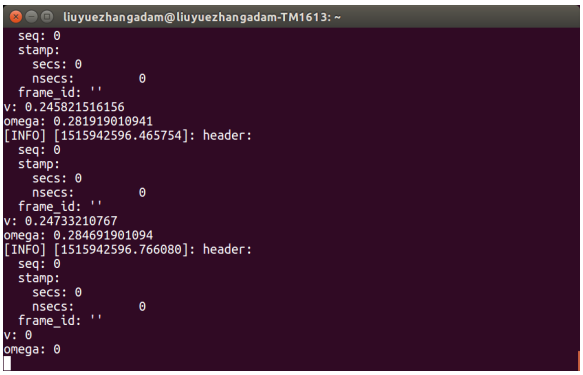

图 10: 运动控制节点 duck_ brain 运行结果

5 总结

本次大作业，是令我收获最大的大作业之一。在此过程中，我学习了新知识；综合运用了计算机网络、数字图像处理与自动控制理论的各方面知识；并且软硬结合，对硬件有了更加深入的体会。

• **ROS 及 Linux 的理解与使用** 实事上，在之前的学习中，我个人已经有了一些 Linux 使用及 ROS 开发的经验。但之前往往停留在读过一边 ROS tutorial 的程度，并没有尝试着自己动手去完成一个基于 ROS 的项目。而通过本次大作业，我比较完整地学习了 ROS 的设计思路，并写具备了订阅、发布消息、自定义消息、修改 CMakeList 和 package.xml 的基础 ROS 开发能力。这是本次大作业我最大的收获。

• **二维码检测** 综合数字图像处理的知识，我尝试了两种二维码检测的方法（在 ros package 的 git log 中可以查到）。一种是本文报告中介绍的二维码角点检测的方法，其优点在于误检率低，近处识别效果好，缺点是漏检率高；另一种方法是采用腐蚀膨胀的方法，相当于检测颜色快速变换的区域，其有点在于漏检率低，远处识别效果好，缺点是误检率较高。实事上，应考虑将两种方法结合使用，依据面积在两种方法中进行切换，以达到远近均可检验的效果。

• **PD 控制及指数平滑** 对于运动控制部分，我分别综合了计算机网络课程及自动控制理论课程中所学的知识，力求达到较好的控制效果。在实际应用中我发现，虽然理论上，这些控制方法具有更优的控制性能，但由于其更复杂，调试起来也比一般方法更加困难，因而导致并未达到明显优于简单时延控制方法的预期效果。即，在不能保证更多的调试时间的情况下，高级的方法并不一定能够保证产生更好的效果。这也让我感受到了理论研究与工程实际的 Gap 所在。

• **硬件调试** 电机等硬件在控制上也存在死区等非理想因素，同时，实验环境中的网络的时延较为严重。各种现实问题，都让我积累了更多宝贵的硬件调试经验。

一言蔽之，学术上，高级、新颖的方法一定是好方法；工程上则不然——只有现场实用的方法，才是工程上的好方法！

综上所述，我强烈地把这个大作业选题推荐给未来的同学。

6 参考文献

1. *ROS Wiki*，Open Source RObotics Foundation，http://wiki.ros.org/ROS/

2. 使用 *OpenCV* 识别 *QRCode*，博客园，https://www.cnblogs.com/jsxyhelu/p/6397118.html

3. *Exponential smoothing*，Wikipedia，https://en.wikipedia.org/wiki/Exponential_ smoothing

4. *py-getch*，joeyespo@github.com，https://github.com/joeyespo/py-getch