# 机器人视觉

技术报告

# 实验一：单线与双线巡线算法实现

## 实验目的

1. 掌握机器人巡线任务中的传感器数据处理方法；

2. 实现PID控制算法对小车方向的动态调整；

3. 完成单线巡线（含S型路径和直角转弯）与双线巡线（含M型路径和环岛处理）任务。

## 实验内容

1. 通过红外传感器或摄像头采集地面黑线位置信息；

2. 编写PID控制算法实现单线巡线；

3. 设计状态机逻辑完成双线巡线中的M型路径跟踪和环岛半圆行驶；

4. 在仿真环境（`task1.world`和`task2.world`）中验证算法有效性。

## 实验器材

- ROS智能车仿真平台（Gazebo环境）

- 红外传感器/摄像头

- Ubuntu 20.04系统（ROS Noetic）

- PID控制器代码模块

## 实验原理

1. 单线巡线：

- \*\*PID控制\*\*：通过比例（P）、积分（I）、微分（D）三环节动态调整小车转向。

- \*\*公式\*\*：

```

```

2. 双线巡线：

- 中线跟踪：根据双线位置计算中线轨迹，控制小车沿中线行驶；

- 状态机设计：将M型路径分解为直行、转弯和环岛处理等状态，通过条件判断切换控制逻辑。

实验步骤

1. 仿真环境配置：

- 将`robot\_task`功能包复制至ROS工作空间`src`目录，执行`catkin\_make`编译；

- 修改`ucar\_robot.launch`文件中的`world`参数，切换`task1.world`或`task2.world`。

2. 单线巡线代码调用：

```bash

# 启动仿真环境

roslaunch robot\_task ucar\_robot.launch

# 运行单线巡线节点

rosrun robot\_task single\_line\_following.py

```

- 关键代码逻辑：

```python

```

3. 双线巡线代码调用：

bash

# 切换至task2.world后运行双线巡线节点

rosrun robot\_task double\_line\_following.py

- 环岛处理逻辑：

```python

if detect\_roundabout():

lock\_steering(angle=1.57) # 逆时针90度转向

drive\_distance(2.0) # 沿半圆行驶

```

实验二：基于YOLOv5的目标检测模型训练与应用

## 实验目的

1. 掌握YOLOv5目标检测环境的配置方法；

2. 训练自定义数据集模型并实现实时检测；

3. 将检测结果通过终端输出。

## 实验内容

1. 在Ubuntu系统中配置YOLOv5依赖环境；

2. 使用标注工具对训练集进行标注，生成`dataset.yaml`；

3. 训练模型并测试摄像头实时检测效果。

## 实验器材

- Ubuntu 18.04系统

- USB摄像头/电脑自带摄像头

- YOLOv5开源代码库

- 标注数据集（COCO格式）

## 实验原理

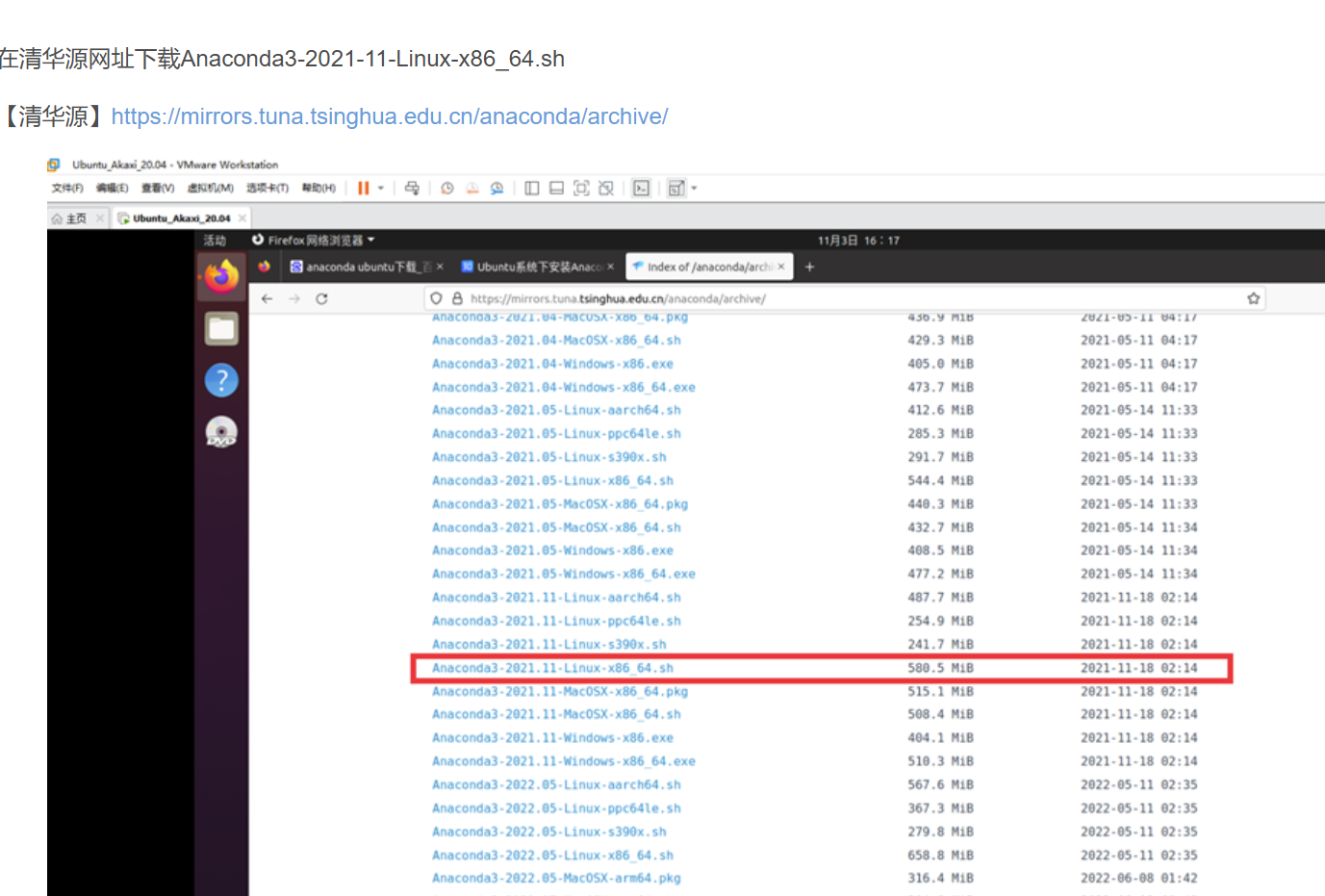
- YOLOv5算法：单阶段目标检测模型，通过锚框（Anchor）机制和特征金字塔网络（FPN）实现多尺度检测；

- 数据增强：采用翻转、裁剪等技术提升模型泛化能力。

## 实验步骤

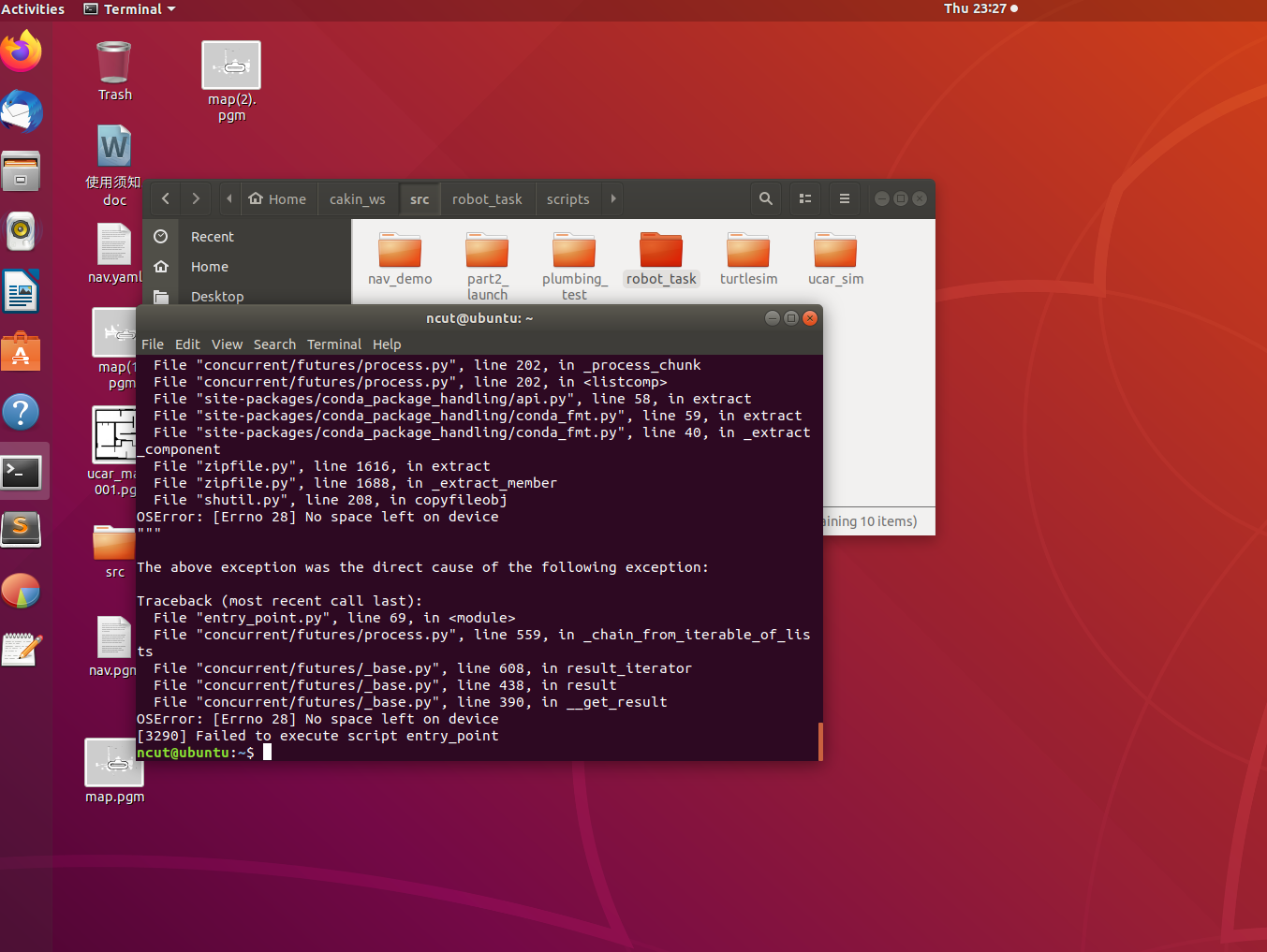
1. 环境配置：

* 首先下载anaconda



* 将下载安装好的包复制到主目录下，并打开终端输入如下命令进行安装

bash Anaconda3-2021.11-Linux-x86\_64.sh



由于虚拟机中ubuntu空间不足，选择在windows系统下进行yolov5环境的配置

```bash

git clone https://github.com/ultralytics/yolov5

cd yolov5

pip install -r requirements.txt

2. 数据集准备：

- 数据集目录结构：

```

dataset/

├── images/

└── labels/

- 生成`dataset.yaml`文件，定义类别和路径。

3. 模型训练：

```bash

python train.py --img 640 --batch 16 --epochs 50 --data dataset.yaml --weights yolov5s.pt

4. 实时检测：

```bash

python detect.py --source 0 --weights runs/train/exp/weights/best.pt

```

- 终端输出示例：

```

Detected: cup, Confidence: 0.92, Position: (320, 240)

```

---

代码运行过程如下

1. 掌握ROS功能包的编译与部署流程；

2. 验证巡线算法在仿真环境中的实际效果；

3. 调试小车初始位置与转向参数。

## 实验步骤

1.功能包部署：

```bash

cp -r robot\_task ~/catkin\_ws/src

cd ~/catkin\_ws && catkin\_make

```

2. 小车初始位姿调整：

- 修改`ucar\_robot.launch`中的`x, y, z`坐标和`Yaw`角度：

```xml

<arg name="x" default="0.0" />

<arg name="yaw" default="1.57" /> <!-- 逆时针90度 -->

```

2. 复现指令：

- 单线巡线：

```bash

roslaunch robot\_task ucar\_robot.launch world:=task1.world

rosrun robot\_task single\_line\_following.py

```

- 目标检测：

```bash

python detect.py --source 0 --weights best.pt

```