**实验指导书**

# 一、实验题目

基于YOLO-OBB算法的交通智能体目标识别算法

# 二、实验目标

本实验基于Ultralytics YOLO框架，通过实现YOLO-OBB旋转目标检测算法，使学生掌握倾斜框检测的核心原理及其在交通场景中的应用。具体目标包括：理解旋转框角度编码与损失计算机制，熟悉DOTA数据集标注规则；利用YOLO框架完成交通目标（如车辆等）的旋转框检测模型训练与部署。

# 三、数据集

## 1. 介绍：

我们数据集包括了68个可用高空鸟瞰图视角视频，其中分为日间时段（38个）以及夜间时段（30个），日间包括5个场景，夜间包括3个场景。

数据集以及附件地址：

链接：https://pan.baidu.com/s/1rsjLBHRqQWTL9QMdalgHDQ?pwd=cowf

提取码：cowf

每一个小组会分配到同一个时段的两个场景的视频，小组成员可以选择其中任意一个场景的视频作为训练用的数据集，另一个作为评判效果的验证数据集。当然，同学们也可以选择补充非所在场景的其他场景的视频来补充自己的，但是分到的两个视频至少需要有一个进行了数据采集。

数据采集流程：  
1. 视频抽帧；

2. LabelImgOBB采集；

3. Dataset-Format-Converter转化文件。

参考抽帧代码：

import cv2

import os

import numpy as np

# 视频路径

D\_train\_path = "your\_video.mp4" # 替换为你的视频路径

# 输出文件夹

output\_folder = os.path.join(os.path.dirname(D\_train\_path), "D\_train")

os.makedirs(output\_folder, exist\_ok=True)

# 打开视频

cap = cv2.VideoCapture(D\_train\_path)

if not cap.isOpened():

print("无法打开视频文件")

exit()

# 获取视频总帧数

frame\_count = int(cap.get(cv2.CAP\_PROP\_FRAME\_COUNT))

# 等距抽取的帧索引

num\_frames\_to\_extract = 20

frame\_indices = np.linspace(0, frame\_count - 1, num\_frames\_to\_extract, dtype=int)

# 开始读取并保存帧

for i, frame\_index in enumerate(frame\_indices):

cap.set(cv2.CAP\_PROP\_POS\_FRAMES, frame\_index)

ret, frame = cap.read()

if not ret:

print(f"无法读取第 {frame\_index} 帧")

continue

# 调整尺寸为1920x1080

resized\_frame = cv2.resize(frame, (1920, 1080))

# 保存图片

output\_path = os.path.join(output\_folder, f"frame\_{i+1:02d}.jpg")

cv2.imwrite(output\_path, resized\_frame)

cap.release()

print("帧提取完成。")

## 2. 目标类别：

|  |  |  |
| --- | --- | --- |
| 标签名 | 含义 | 备注 |
| pedestrian | 行人 |  |
| two-wheeler | 两轮交通工具（含人力、电动或燃油驱动） | 包括自行车、电动自行车、摩托车等； |
| three-wheeler | 三轮交通工具 | 包括正三轮（后两轮）、倒三轮（前两轮）、侧倾三轮（单侧两轮，另一侧单轮）； |
| small-vehicle | 小型车辆 | 包括轿车、SUV、MPV等； |
| bus | 公交车 | 含城市公交、长途巴士、校车等； |
| articulated-truck-tractor | 分离式卡车牵引车（动力部分） | 即“卡车头”； |
| articulated-truck-trailer | 分离式卡车挂车（无动力货箱） | 即“卡车车厢”； |
| rigid-truck | 一体式卡车（车头与货箱固定连接） |  |
| other | 其他交通工具 | 包含工程机械（挖掘机、推土机）等其他车辆； |

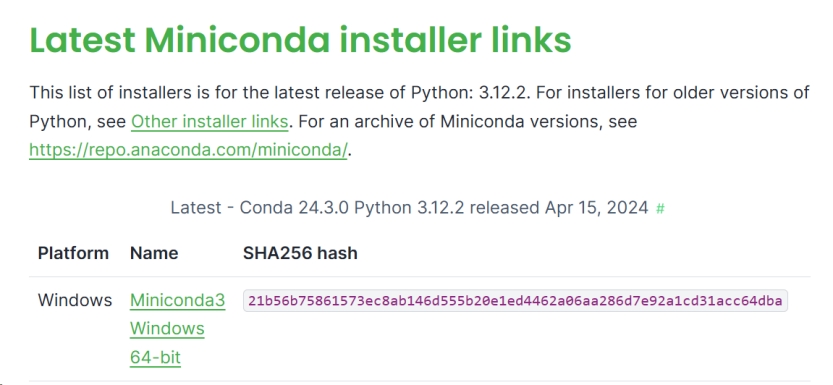
# 四、实验步骤

## （1）软件安装和环境配置

安装miniconda（若已安装anaconda/miniconda可跳过）

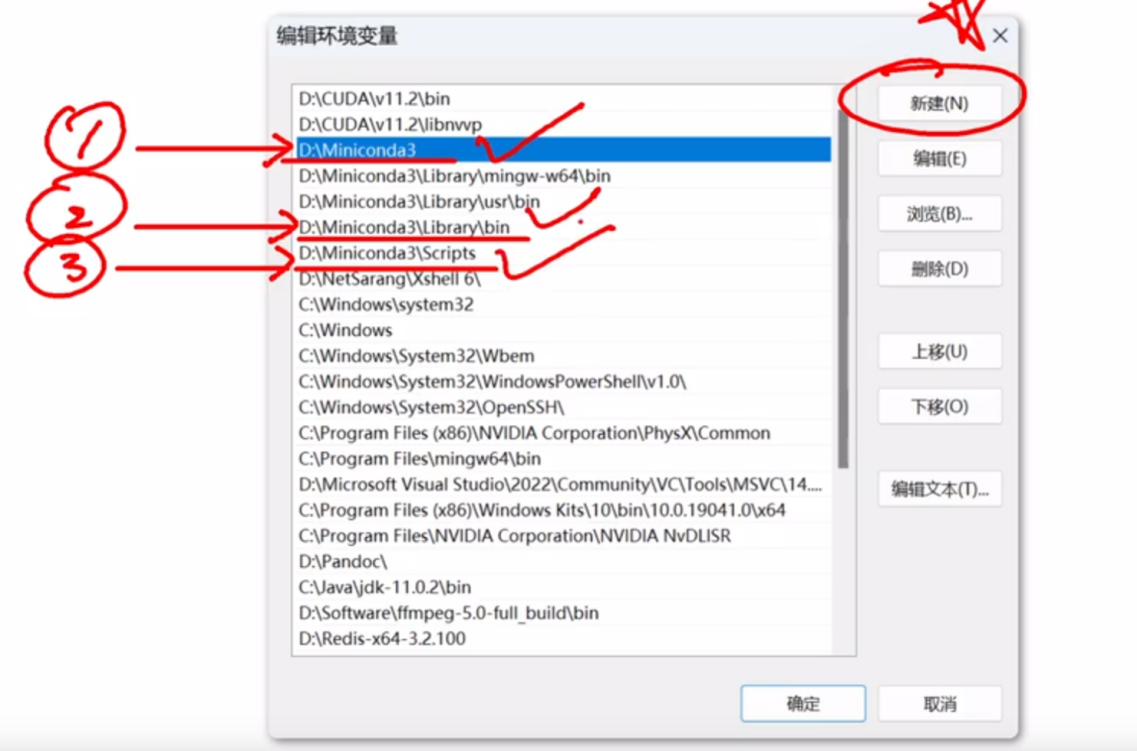
<https://docs.anaconda.com/free/miniconda/index.html>

选择windows版本下载



安装过程及环境变量设置：

https://www.bilibili.com/video/BV1mb41197Pr/?p=2&vd\_source=e9f767076d5ef664054c4bfd4eecc651



创建并激活新的环境

在开始菜单选择命令行工具



## （2）创建LabelIMG OBB运行环境

通过以下链接下载数据采集软件LabelImg\_OBB版本

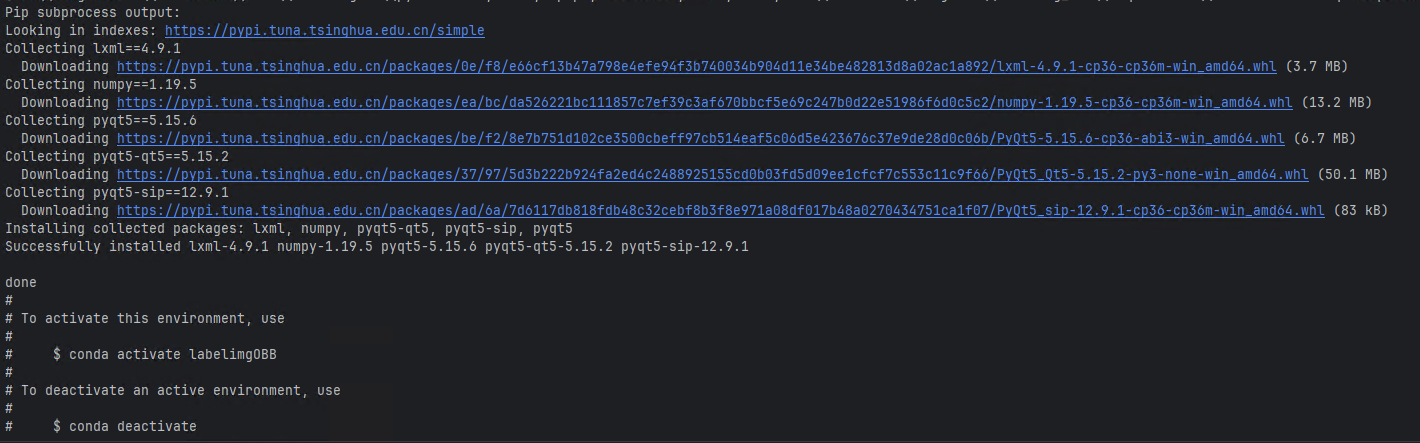
<https://github.com/BIANG-qilie/labelimg_OBB>

通过以下链接下载LabelImg\_OBB 转换为 DOTA格式软件 （这两个项目都可以star一下，有问题可以提issue）

https://github.com/BIANG-qilie/Dataset-Format-Converter

解压后，进入LabelImg\_OBB 文件夹，输入创建环境的命令：

conda env create -f requirement/labelimgOBB.yml



如果需要确认则输入y并回车确认，安装成功后启动：

conda activate labelimgOBB

A black screen with white text

AI-generated content may be incorrect.

dataset-format-converter则根据github说明文档安装即可：  
pip install dataset-format-converter

## （3）LabelIMG\_OBB工具使用

进入LabelImg\_OBB 文件夹下，输入以下指令启动脚本：

python labelImg.py

A screenshot of a computer

AI-generated content may be incorrect.

进入软件后可以点击左上角File，点击Preferences选择常用的语言。

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

下面以简体中文为例，操作步骤大致如下：

1. 点击打开目录 ,并选择即将采集的图片所在的文件夹；
2. 通过CRTL+➕、CRTL+➖或者CRTL+滚轮调整画布大小，使用快捷键W,创建一个旋转框，并选择或输入一个标签名，点击OK。

A screenshot of a computer

AI-generated content may be incorrect.

1. 调整方框角度，方式有两种：
   1. 选中方框中的其中一个点，右击拖动即可旋转；
   2. 选中方框，按下以下按钮进行旋转，O-顺时针旋转0.1度, P-逆时针旋转0.1度, K-顺时针旋转1度, L-逆时针旋转1度, M-顺时针旋转5度, ，逆时针旋转5度（可通过帮助tab->快捷键说明进行查看）。

A screenshot of a computer screen

AI-generated content may be incorrect. A screen shot of a computer

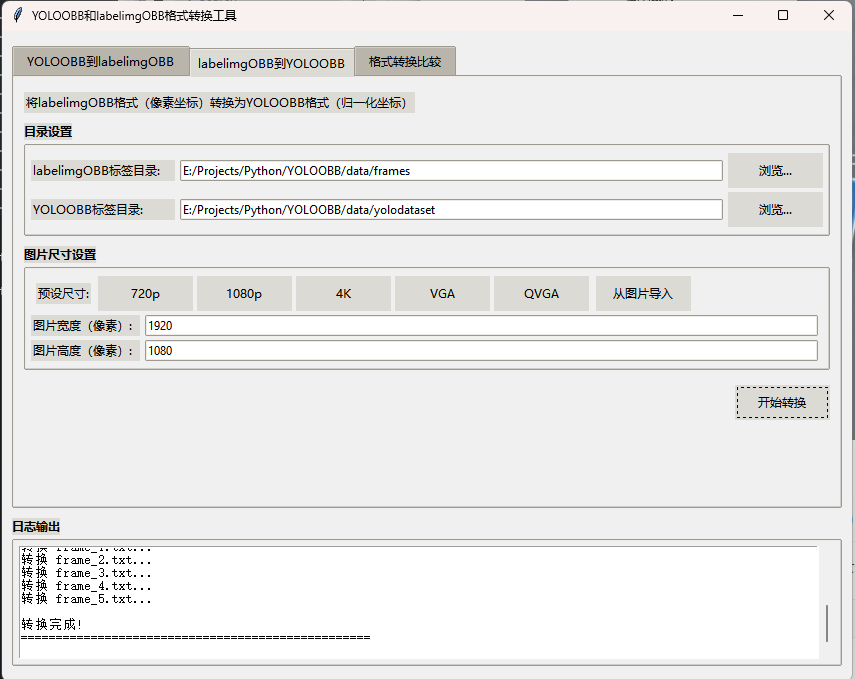
AI-generated content may be incorrect.

1. 点击保存，或者使用CTRL+S 保存当前标注txt。
2. 在操作结束之后，点击右上角关闭键进行关闭（Windows）。

## （4）Dataset-Format-Converter工具使用

进入Dataset-Format-Converter文件夹下，输入以下指令启动脚本(沿用labelimgOBB环境)

python convert\_labels\_gui.py



1. **点击labelimgOBB到YOLOOBB 这个tab；**
2. 选择采集好数据的目录labelimgOBB标签目录以及预计要输出的目录YOLOOBB标签目录；
3. 输入图片尺寸；
4. 点击开始转换。

## （5）YOLO obb数据集构建

### （5.1）目录结构

your\_dataset/   
├── images/ # 存放所有图像文件   
│ ├── train/ # 训练集图像   
│ ├── val/ # 验证集图像   
│ └── test/ # 测试集图像   
├── labels/ # 存放所有标签文件（与images目录一一对应）   
│ ├── train/   
│ ├── val/   
│ └── test/   
├── class\_names.txt # 类别名称列表文件 ，Dataset-Format-Converter工具会导出  
└── dataset.yaml # YAML配置文件，Dataset-Format-Converter工具会导出

### （5.2）图像文件（images文件夹）

1. **格式要求**：建议统一为 .jpg 或 .png。
2. **命名规则**：图像与标签文件需同名（如 001.jpg 对应 001.txt）。
3. **数据划分**：建议按比例分割为 train/val/test 子目录（比例例如 70%/20%/10%）。

### （5.3）标签文件（labels文件夹）

1. **文件格式**：每个图像对应一个 .txt 文件，格式如下：

<class\_id> <x1\_center> <y1\_center> <x2\_center> <y2\_center> <x3\_center> <y3\_center> <x4\_center> <y4\_center>

1. **归一化坐标**：所有值需归一化为 0~1（相对于图像宽高）。
2. **示例**：

4 0.184855 0.931481 0.231730 0.931481 0.231730 0.964815 0.184855 0.964815

### （5.4）类别文件（class\_names.txt），Dataset-Format-Converter工具会导出

* **格式**：按类别编号顺序逐行写入类别名称。
* pedestrian   
  two-wheeler   
  three-wheeler   
  small-vehicle   
  bus   
  articulated-truck-tractor   
  articulated-truck-trailer   
  rigid-truck   
  other

### （5.5） YAML配置文件（dataset.yaml），Dataset-Format-Converter工具会导出，但是path字段需要修改：

# 数据集路径配置   
path: ./your\_dataset # 数据集根目录   
train: images/train # 训练集路径（相对path）   
val: images/val # 验证集路径   
test: images/test # 测试集路径（可选）

nc: 9   
# 类别信息   
names: ['pedestrian', 'two-wheeler', 'three-wheeler', 'small-vehicle', 'bus', 'articulated-truck-tractor', 'articulated-truck-trailer', 'rigid-truck', 'other']

关键注意事项

1. **标签一致性**：确保每个图像有对应的标签文件，且文件名严格匹配。
2. **类别编号**：class\_names.txt 和 YAML文件中的类别顺序必须一致。

## （6）YOLO obb环境配置

使用下列指令配置，其中yoloobb.yml 为附件中的配置环境的路径

conda env create -f yoloobb.yml

如果需要确认则输入y并回车确认，安装成功后启动

conda activate yoloobb

其中的torch环境是基于cuda12.6的版本，如果和你的cuda版本和我的一致，可以继续使用，如果不一致，则需要卸载掉当前的torch版本，指令如下：

pip uninstall torch torchaudio torchvision

可以前往<https://pytorch.org/get-started/locally/>找寻适合自己的版本，也可以使用通用的cpu版本

pip install torch torchvision torchaudio

## （7）YOLO obb训练

创建一个脚本，并根据https://docs.ultralytics.com/modes/train/#usage-examples编写训练脚本，以下是参考:

from ultralytics import YOLO

if \_\_name\_\_ == '\_\_main\_\_':

    # Load a model

    model = YOLO("yolo11n-obb.pt").to("cuda:0")  # load a pretrained model (recommended for training)

    results = model.train(

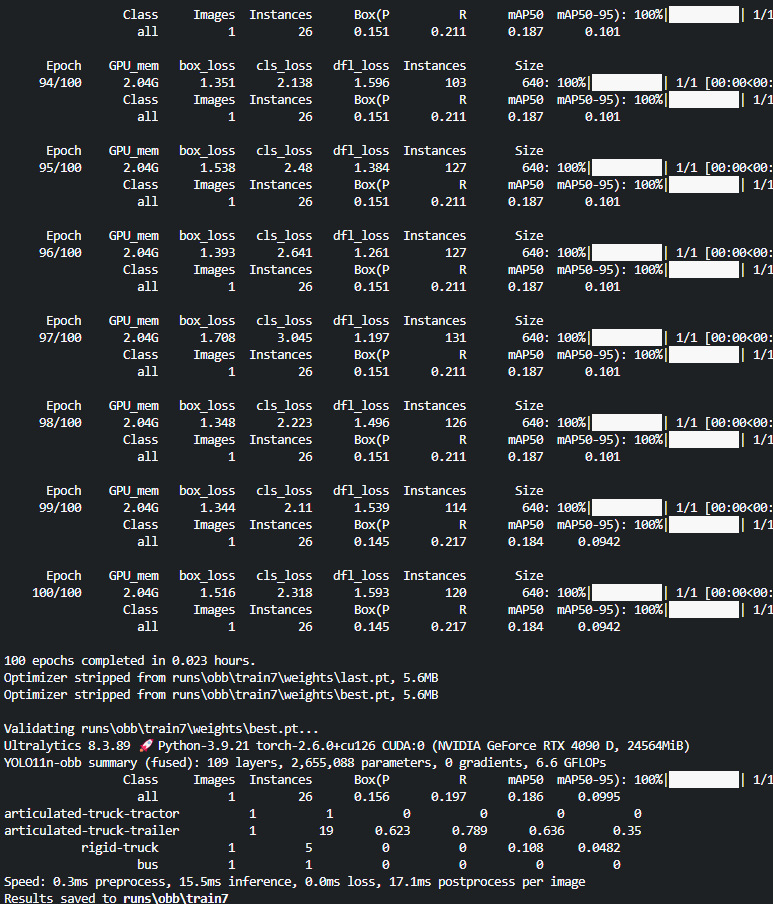
            data=r"E:\Blake Zhu\Projects\YOLO11\data\yoloobb\_dataset\dataset.yaml",# your dataset.yaml

            epochs=100

        )

其中预训练模型yolo11n-obb.pt可以在[https://docs.ultralytics.com/tasks/obb/#visual-samples](https://docs.ultralytics.com/tasks/obb/" \l "visual-samples)中下载；

运行脚本后会对dataset生成cache文件，方便对数据进行读取，并进行训练。



训练结果默认会保存在runs文件夹下。

## （8）YOLO obb预测

创建一个脚本，并根据https://docs.ultralytics.com/modes/predict/编写训练脚本，以下是参考:

from ultralytics import YOLO

if \_\_name\_\_ == '\_\_main\_\_':

    # Load a model

    model = YOLO("yolo11n-obb.pt").to("cuda:0")  # 换成训练好的模型

    results = model("data/DJI\_20230427182506\_0001\_W.MP4",    # 选择D\_val的视频

                          save=True,

                          save\_txt=True,    # 保存检测框坐标到txt文件

                          save\_conf=True,   # 在txt中包含置信度

    )

运行结束后会显示结果所在的位置，也可以试着在加入show参数来实现展示。

## （9）YOLO obb跟踪预测

创建一个脚本，并根据https://docs.ultralytics.com/modes/track/#available-trackers编写训练脚本，以下是参考:

from ultralytics import YOLO

if \_\_name\_\_ == '\_\_main\_\_':

    # Load a model

    model = YOLO("yolo11n-obb.pt").to("cuda:0")  # 换成训练好的模型

    results = model.track("data/DJI\_20230427182506\_0001\_W.MP4"

,    # 选择D\_val的视频

                          save=True,

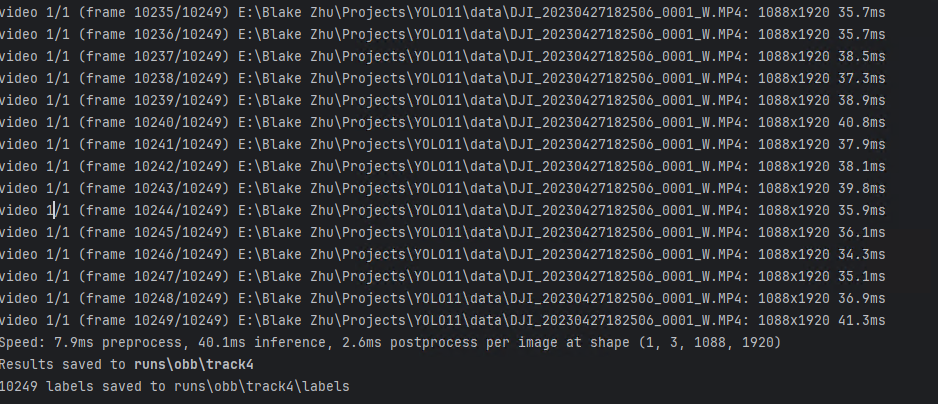
                          save\_txt=True,    # 保存检测框坐标到txt文件

                          save\_conf=True,   # 在txt中包含置信度

    )

运行结束后会显示结果所在的位置，也可以试着在加入show参数来实现展示。

当参数设置为save=True，save\_txt=True，save\_conf=True时，会在结果所在的位置的labels文件下保存每一帧的跟踪预测结果（该部分需要作为实验结果提交），如下图就位于“runs\obb\track4\labels”。



# 五、提交要求

（1）1-5人为一组进行实验，以小组为单位，提交以下实验结果：

a、Dataset-Format-Converter工具选择的LabelImgOBB标签目录下的内容（包括 classes.txt文件、图片文件、与同名的txt标签文件）；

b、每一帧的跟踪预测结果（在第四点的第9小点有提到怎么获取）；

c、源代码；

d、实验报告（每组可合作写，但是每人都需要交一份，文件命名格式为：个人班级+学号+姓名）；

e、识别和跟踪效果视频展示（可选）。

（2）各班学委收集各组上述文档（以组为单位放在同一文件夹），最后打包为1个文件提交邮箱（weiliangzeng49@163.com）。

（3）所有资料提交截止日期：2025年7月10日