自动网球识别机器人实验报告

1. 实验背景

随着体育训练智能化的发展，传统人工捡拾网球的方式逐渐暴露出效率低、重复劳动强度大等问题，严重影响训练节奏与体验。为了提升训练自动化水平，自动捡球机器人应运而生，其核心任务之一便是能够精准、高效地识别并定位散落在场地上的网球。本项目围绕“网球自动捡球机器人”展开，旨在利用计算机视觉与深度学习目标检测技术，对网球场景中的网球进行实时识别与定位。检测信息将作为后续路径规划与控制执行的基础输入，直接影响整个系统的性能与稳定性。

本实验选用当前成熟的 YOLOv8（You Only Look Once Version 8）目标检测算法，构建端到端的网球识别模型。通过优化模型结构、改进损失函数与训练策略，进一步提升其在小目标检测场景下的鲁棒性与准确性，并验证其在实际场地条件下的应用效果。

2. 实验环境与数据集

本次实验在 Windows 10 操作系统的 PC 上进行，硬件包括 Intel i3-10100F 四核处理器和 NVIDIA GeForce RTX 4060 8GB 显卡。深度学习平台基于 PyTorch 和 Ultralytics 提供的 YOLOv8 框架构建。实验中使用了 YOLOv8 的预训练模型权重文件 yolov8n.pt，该模型为 YOLOv8 的 nano 版本，体积小、推理速度快，特别适合在本次配置的普通桌面硬件环境下进行高效训练与测试。

本实验使用的网球目标检测数据集结构遵循 YOLO 格式。训练集包含 830 张图像，验证集包含 256 张图像，其中一部分图像来源于网络收集，另一部分为成员使用手机实拍并手工标注的图像。手动标注使用 YOLO 格式，每张图像对应一个包含类别及相对坐标的 TXT 文件。为了提升模型泛化能力，在训练前对原始数据进行了本地静态增强处理。本地静态增强操作通过自定义脚本完成，使用了 Albumentations 库实现以下几种增强方式：亮度/对比度调整、水平翻转、平移缩放旋转变换（ShiftScaleRotate）、模糊处理等。每张图像增强三次，增强图

像及对应标注被保存到专用目录，并与原始数据共同用于模型训练。总体而言，数据集涵盖了网球在不同光照、角度和背景下的外观，以提高模型的鲁棒性。

此外，数据集配置文件 data.yaml 已正确配置训练和验证图像、标签路径，以及类别数量与名称。

图 2.1 数据配置

3. 实验方案

3.1.模型结构

本实验选用 YOLOv8 目标检测模型。YOLOv8 延续了 YOLO 系列单阶段检测器的设计，其网络由 Backbone（骨干特征提取网络）、Neck（特征融合网络）和 Head（检测头）组成。YOLOv8 的 Backbone 采用了 CSP 结构改进的 C2f 模块以及 SPPF 快速池化层来提取多尺度特征；Neck 部分使用 FPN/PAN 结构融合不同层次的特征。相比前代 YOLOv5，YOLOv8 在 Head 部分做出了重要改进：采用 Anchor-Free 的设计（摒弃了预设锚框，直接预测目标的中心点坐标及宽高），并引入了解耦头（Decoupled Head）结构，将分类分支和回归分支分离，从而减少任务冲突，提高检测精度。这种设计减少了锚框超参数的复杂性，同时提升了模型对复杂场景的适应性。此外，YOLOv8 使用了 CIoU 损失等改进的损失函数，提高边界框回归的精度，并采用了动态标签分配等策略优化正负样本分配。实验中使用的 yolov8n 是 YOLOv8 系列中最小的模型（nano 版本），约有数百万级别参数，模型尺寸小但仍具有良好的性能，适合在资源受限环境下实时检测。

3.2.实验主要训练超参数设置

Batch size（批大小）：设置为 16。考虑到本地训练硬件的实际显存容量（RTX 4060 8GB），选择较为适中的批大小以兼顾训练稳定性与内存负载。该设置在当前配置下能够高效运行并充分利用 GPU 资源。

学习率（learning rate）：初始学习率设为 0.0015，使用 AdamW 优化器时较小的学习率有助于训练过程更为稳定。Ultralytics 框架会自动使用余弦退火（Cosine Annealing）等调度策略在训练过程中动态调整学习率，实现先升后降的学习率调度过程，从而在后期获得更细致的权重更新。

优化器（optimizer）：采用 AdamW 优化器。该优化器具备良好的自适应性，并在训练中施加权重衰减，有效防止模型过拟合，适合较长周期训练。相较于默认的 SGD，AdamW 无需手动设定动量参数，且在本实验中表现出更稳定的收敛趋势。

训练轮数（epochs）：设置为 500 轮。较长的训练周期保证模型有充分时间进行参数拟合，提升性能。训练过程中设定早停策略（patience=100），即若验证指标连续 100 轮无提升，将提前终止训练以节省资源。

输入图像尺寸（image size）：采用 多尺度训练，图像输入尺寸在 [512, 736] 区间随机变化。YOLOv8 支持自适应填充与缩放，该策略可模拟目标在不同距离和视野下的分布，提升模型对不同尺度网球的适应能力。相较于固定尺寸输入，多尺度训练更具泛化性，尤其适合实际场景中目标尺寸多变的检测任务。

3.3.动态数据增强

启用多种增强方式，包括马赛克（mosaic=0.8）、混合增强（mixup=0.2）、色彩扰动（hsv_h/s/v）、仿射变换（degrees=10, scale=0.8）等。这些增强策略通过 Ultralytics 参数接口指定，并在训练脚本中结合 albumentations 进一步扩展，增强模型对不同场景与目标形态的适应性。训练前 10 轮使用较强增强（close_mosaic=10），之后逐步降低，以便后期模型细化学习。

3.4.训练流程

本实验基于 Ultralytics YOLOv8 框架，对轻量化模型 YOLOv8n 进行微调。训练初始化时加载 COCO 预训练权重 yolov8n.pt，并结合自定义结构文件 yolov8n_p2.yaml 以提升模型在网球检测任务中的表现。

训练过程通过 .train() 接口调用 data.yaml 文件加载自定义数据集，采用 batch size = 16，训练轮数设定为 500。为增强模型的泛化能力，训练过程中启用了多种数据增强策略，包括亮度/饱和度扰动、随机旋转、缩放、HSV 色

彩扰动、MixUp、Mosaic 与 Copy-Paste 等。

增强后的图像输入模型主干网络进行特征提取，经由特征融合模块与检测头，输出检测结果。训练采用 AdamW 优化器，并启用 AMP 混合精度以加速训练、降低显存使用。GPU 显存占用约为 3.63GB，资源占用较低，适合中端显卡训练部署。训练过程共设定最大 500 个 epoch，但模型在第 469 个 epoch 达到最优验证性能，具体指标如下（mAP@0.5 = 0.982，mAP@0.5:0.95 ≈ 0.8316），训练过程稳定，性能显著提升，并将该 epoch 的模型权重保存为 best.pt。

3.5.项目配置步骤

首先克隆项目代码：

Git clone https://github.com/lwhkkkk/object-detection-competition.git

然后在项目根目录下打开终端，执行以下命令安装依赖：pip install -r requirements.txt

在安装依赖并准备好输入图像后，直接运行以下命令即可进行目标检测：python main.py

程序将自动读取 src/images/ 目录中的图像，进行网球目标检测，并将检测结果保存至：output/output_results.json：检测框及坐标等结构化结果；

项目结构说明

```
project-root/
├── main.py              # 主程序入口，执行推理与保存
├── process.py           # 模型推理逻辑
├── models/
│   └── best.onnx        # 训练好的 YOLOv8 ONNX 模型
├── src/
│   └── images/          #  测试图像请放在此目录
├── output/
│   ├── output_results.json  # 所有检测结果 JSON 输出
│   └── images/              # 带检测框的图像输出（如果想要图像输出可将代码图像输出代码取消注释）
├── requirements.txt     # 项目依赖列表
└── README.md            # 本说明文档
```

图 3.1 项目结构说明

4. 模型优化

4.1.DIOU 算法优化

为提升本项目在网球小目标检测任务中的定位精度和收敛效率，我们在目标

检测模型的边界框回归损失函数中，引入了 DIoU（Distance Intersection over Union）损失替代默认的 CIoU（Complete IoU）损失。

4.1.1. DIOU 损失函数

DIoU 是一种基于 IoU 的改进型距离度量方法，其在计算边界框回归损失时，不仅考虑预测框与真实框之间的重叠程度（IoU），还引入了中心点之间的欧式距离作为惩罚项。其数学形式如下：

$$\mathrm{DIOU} = 1 - \mathrm{IoU} + \frac{\rho^2(b, b_{gt})}{c^2}$$

4.1.2. DIOU 损失函数的优点

提升定位精度：DIOU 在计算 IoU 的基础上引入了预测框与真实框中心点之间的距离惩罚项，使得模型在训练过程中不仅关注框重叠面积，还能更快速地将预测框向目标中心对齐，从而提升边界框的定位准确性，尤其适用于小目标检测。

加快收敛速度：由于 DIOU 考虑了几何位置的收敛，引导模型更精准地逼近目标中心，因此相比 CIoU 等损失，通常能够在较少的 epoch 内达到较优精度，减少训练时间。

适应复杂场景下的检测需求：在遮挡、目标密集或目标较小等复杂场景中，DIOU 对目标中心位置的建模优势尤为明显，可以显著减少预测框偏移导致的误检与漏检。

4.1.3. DIOU 损失函数的实现

```
    def forward(
        self,
        pred_dist: torch.Tensor,
        pred_bboxes: torch.Tensor,
        anchor_points: torch.Tensor,
        target_bboxes: torch.Tensor,
        target_scores: torch.Tensor,
        target_scores_sum: torch.Tensor,
        fg_mask: torch.Tensor,
    ) -> Tuple[torch.Tensor, torch.Tensor]:
        """Compute IoU and DFL losses for bounding boxes."""
        weight = target_scores.sum(-1)[fg_mask].unsqueeze(-1)
        # iou = bbox_iou(pred_bboxes[fg_mask], target_bboxes[fg_mask], xywh=False, CIoU=True)

        iou = bbox_iou(pred_bboxes[fg_mask], target_bboxes[fg_mask], xywh=False, DIoU=True)
```

图 4.1 DIOU 损失函数代码

4.2.引入 P2 输出分支

由于网球属于典型的小目标，在图像中的尺寸较小，容易在深层特征中被弱化，YOLOv8n 默认仅使用 P3（下采样 8 倍）、P4（16 倍）、P5（32 倍）进行检测，难以有效感知更小的目标。为提升对小网球的感知能力，我们小组在 YOLOv8n 的基础上引入了 P2 检测分支（下采样 4 倍），增强模型对浅层细节特征的利用能力，从而提升小目标检测效果。

4.2.1. P2 输出分支实现方式

在 Ultralytics YOLOv8n 的基础上，首先通过自定义模型结构，引入了一条从第 2 层输出（P2 层）派生出的高分辨率检测路径，以增强模型对小目标的检测能力。具体实现包括：在 head 中添加一条由 Upsample + Concat + C2f 组成的分支，用于提取和融合来自浅层（P2）的特征；修改原始 Detect 层的输出结构，将 [P3, P4, P5] 调整为 [P5, P4, P2]，使模型更加优先利用高分辨率的特征图进行小目标检测；同时保持 YOLOv8n 原有的轻量化配置，即 depth_multiple=0.33 与 width_multiple=0.25，确保模型在边缘设备上的部署效率；最后，将类别数 nc 设置为 1，以适应本项目中的网球单类别检测任务。

4.2.2. P2 输出分支实现代码

```
    [-1, 1, Conv, [64, 1, 1]],
    [-1, 1, nn.Upsample, [None, 2, "nearest"]],
    [[-1, 2], 1, Concat, [1]],      # ✅ cat P2
    [-1, 3, C2f, [64]],             # ✅ 新增 P2 head
    [[10, 13, -1], 1, Detect, [nc]], # Detect(P5, P4, P2)
]
```

图 4.2 P2 输出分支实现代码

4.2.3. 优化效果

通过本次模型结构优化，引入 P2 输出分支后，模型在小尺寸目标检测任务中的性能得到了提升。首先，在图像中包含远距离或体积较小的目标（如网球）时，模型对其的识别能力明显增强，检测精度与召回率均有提高；其次，该结构在保持模型轻量化的同时，适配边缘设备的算力限制，依然能够实现 mAP50 和 mAP50-95 等精度指标的提升；最后，优化后的模型减少了漏检与误检的情况，在实际推理中展现出更强的鲁棒性和泛化能力。

4.3.训练方式优化

4.3.1. 多尺度训练策略

为提升模型对不同尺寸图像中目标的适应能力，采用［512，736］范围内的多尺度训练配置，使模型能够同时学习小图细节与大图全局特征，从而增强对远近不同场景下网球的检测鲁棒性。

```
patience=100,
imgsz=[512, 736],  # ✅ 多尺度训练
batch=16,
device=0,
```

图 4.3 多尺度训练策略代码

4.3.2. 动态数据增强方法优化

在默认增强基础上，启用了如 Copy-Paste（0.3）、Mosaic（0.8）、MixUp（0.2）等策略，模拟网球在复杂背景、密集排布或遮挡等情形下的出现。同时添加了图像扰动增强（如：亮度、对比度、色调变化及旋转缩放等），提升模型泛化能力与抗干扰能力。

```
# ✅ 图像增强
degrees=10,
scale=0.8,
hsv_h=0.08,
hsv_s=0.7,
hsv_v=0.5,
mosaic=0.8,
mixup=0.2,
copy_paste=0.3,  # ✅ 新增 Copy-Paste 增强
```

图 4.4 动态数据增强优化代码

4.3.3. 损失函数与优化器调整

训练时使用了 AdamW 优化器以增强在小数据集下的稳定性；设置 label_smoothing=0.05 减缓过拟合，适配 single_cls=True 以专注于网球目标的单类别学习任务。通过适度调低 box=0.05 和调高 cls=0.5 的损失比重，引导模型更加注重目标定位与分类。

```
    workers=2,
    optimizer="AdamW",
    lr0=0.0015,
    lrf=0.01,
    warmup_epochs=3,
    single_cls=True,
    label_smoothing=0.05,
    close_mosaic=10,
    rect=True,
    amp=True,
    overlap_mask=False,
    augment=True,  # ✅ 启用内建增强策略
    box=0.05,
    cls=0.5,
```

图 4.5 损失函数与优化器调整代码

4.3.4. 自动记录与实验管理

训练脚本自动记录时间戳、模型结构、评估指标（mAP50 与 mAP50-95）等关键信息，便于后续性能分析与模型版本对比。

```
with open(os.path.join(project_dir, "training_info.txt"), "w") as f:
    f.write(f"模型结构: {model_weight}\n")
    f.write(f"数据集: {dataset_yaml}\n")
    f.write(f"训练耗时: {(time.time() - start_time) / 60:.2f} 分钟\n")
    f.write(f"mAP50: {val_results.box.map50:.4f}\n")
    f.write(f"mAP50-95: {val_results.box.map:.4f}\n")
```

图 4.6 自动记录与实验管理代码

5. 实验结果展示

5.1.结果分析

模型训练共进行 500 轮，整体收敛良好。下图展示了训练与验证过程中的关键指标变化情况：

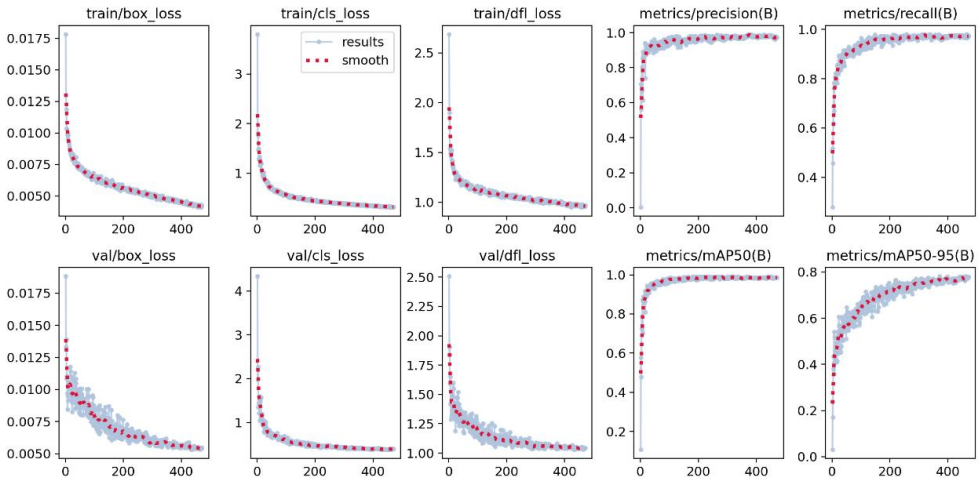

图 5.1 模型训练过程中的损失变化与精度指标曲线图

损失函数收敛情况：训练与验证阶段的 box loss、cls loss 和 dfl loss 均快速下降并趋于稳定，表明模型已有效学习目标的边界框、分类及分布信息。其中 box_loss 最终稳定在 0.005 左右，cls_loss 降至 0.5 以下，反映出模型具备良好的回归与分类能力。

精度指标表现：模型在训练中后期的 Precision 和 Recall 均超过 0.95，说明模型在检测网球目标时漏检率和误检率极低，检测稳定性好。

mAP 提升情况：mAP@0.5 最终达到 0.98+，说明模型对目标是否存在的判断非常准确；mAP@0.5:0.95 达到 0.82+，在更高 IoU 阈值下依然保持优异表现，验证了模型的定位精度和鲁棒性。

整体来看，训练过程无过拟合迹象，模型在多尺度、复杂背景和小目标任务中具备稳定、优越的检测能力。

5.2.可视化分析

5.2.1. 混淆矩阵（Confusion Matrix）

该混淆矩阵表明，模型在网球检测任务中具备较高的分类准确率与鲁棒性：真正例（TP）为 367，说明绝大多数实际存在的网球都被正确识别；假正例（FP）为 37，即背景被误识别为网球的情况较少；假负例（FN）仅为 7，表示漏检率极低，模型对目标几乎无遗漏。从整体趋势来看，模型在 item（网球）类别上具有良好的检测能力与容错能力，并有效控制了误检与漏检。该结果也与 mAP50、recall 指标保持一致，进一步验证了模型在实际场景下的有效性与稳定性。

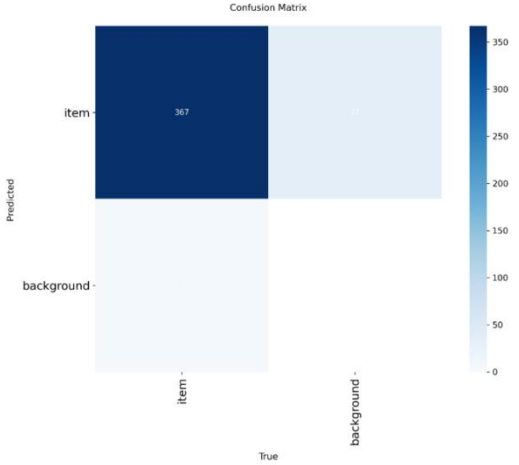

图 5.2 模型在验证集上的混淆矩阵（Confusion Matrix）

5.2.2. F1-Confidence 曲线

该图展示了模型在不同置信度阈值下的预测表现。可以观察到：当置信度较低（<0.3）时，模型更倾向于召回（Recall）更多目标，但误检较多，导致 F1 分数偏低；当置信度过高（>0.85）时，模型虽减少了误检，但也错过了部分目标（召回下降），F1 同样下降；在置信度为 0.587 附近时，模型达到最佳 F1 得分 0.96，说明此时 Precision 与 Recall 达到最优平衡。这一分析对于后续推理阶段的阈值选择具有参考价值 —— 推荐将检测阈值设置为 0.58 左右，以获得最佳检测效果。

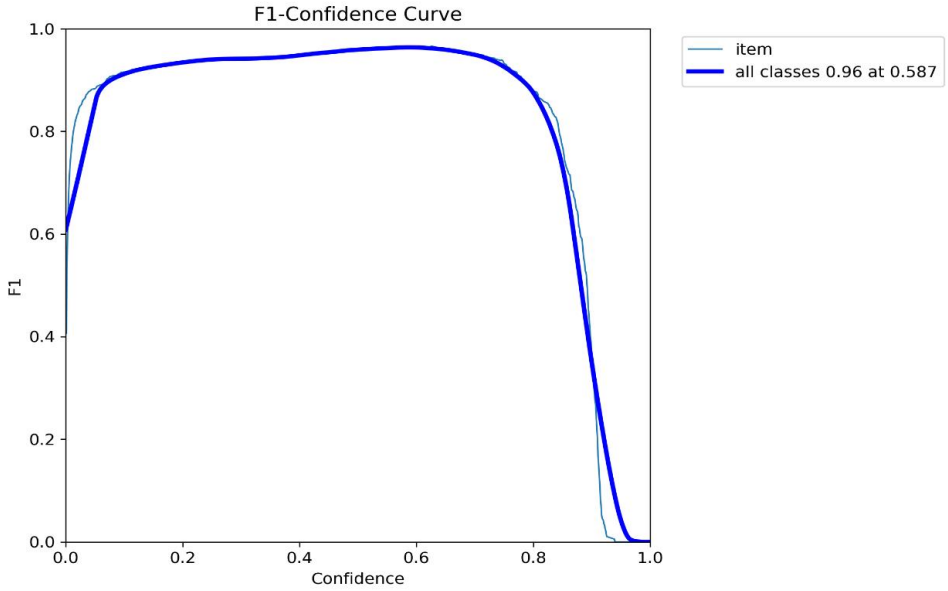

图 5.3 F1-Confidence 曲线

5.2.3. 精度-召回率曲线

图中展示的是模型在不同置信度阈值下，精度与召回率之间的变化关系：曲线整体接近右上角，说明模型在大多数置信度阈值下都能保持高精度和高召回；mAP@0.5 达到 0.988，表明在 IoU ≥ 0.5 的条件下，模型在目标识别与定位方面具备极高的准确性；Precision 在 Recall 提升时下降幅度极小，说明模型在不牺牲召回的前提下，也能有效控制误检率。整体而言，模型 P-R 表现优异，尤其适合对漏检容忍度低、需高召回率的实际检测场景，如本项目中的小目标网球检测任务。

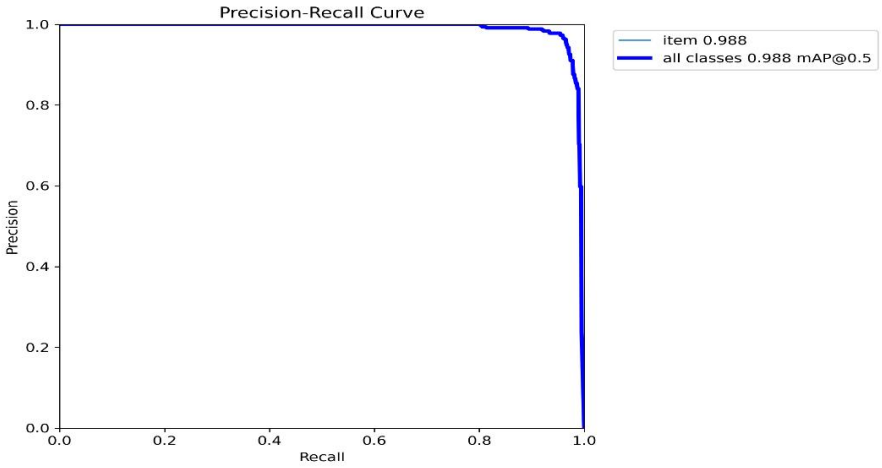

图 5.4 Precision-Recall 曲线图

5.2.4. 数据分析可视化图

该图展示了训练集中 item 类（网球）的样本结构特征：

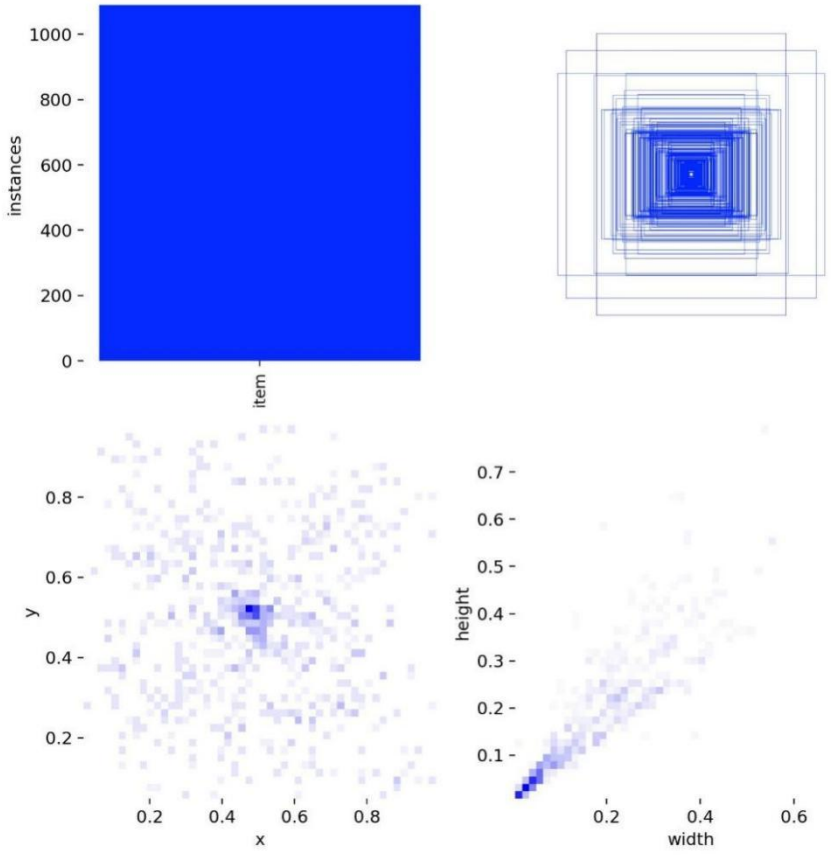

图 5.5 数据集目标分布可视化

样本数量充足：item 类样本数接近 1000，满足训练需求，无明显类别失衡；

目标尺寸分布集中：大部分网球尺寸较小且较为统一，便于模型进行 anchor 聚类与尺度学习；

空间分布集中：中心点分布主要集中于图像中间区域，表明拍摄画面相对规范，有利于模型收敛；

尺寸密集小目标特征明显：右下角分布图揭示了这是一个标准的小目标检测任务，后续优化（如添加 P2 分支）具备明确意义。

5.3.项目亮点

本项目在模型设计与优化方面具有以下突出亮点:

模型极度轻量化：使用 YOLOv8n（nano 版）作为基础模型，并在其上做小幅结构增强（添加 P2 分支），最终导出的 ONNX 模型仅 6.69MB，大幅优于常规检测模型，适合在如飞腾派等资源受限设备上快速部署；

运行效率极高：桌面端实测平均推理时间约 91ms/张图像，，具备准实时处理能力，能很好支撑捡球机器人对实时性的要求；

推理性能与检测精度兼得：通过引入浅层检测分支与自定义损失函数，显著提升了小目标检测能力，同时保持优良的 mAP 表现；

可扩展性强：支持量化、剪枝、注意力机制集成等后续优化，为边缘部署与多场景迁移奠定基础。

6. 总结与展望

6.1.总结

本实验基于 YOLOv8n 目标检测框架，围绕网球小目标检测任务，完成了从数据准备、模型训练到结构优化与结果验证的完整流程。通过引入 P2 输出分支、使用 DIoU 损失函数、多尺度训练与动态数据增强等策略，有效提升了模型对小目标的检测精度与鲁棒性。最终模型在验证集上取得了 mAP@0.5 = 98.8%、mAP@0.5:0.95 ≈ 83.16%的优异表现，训练过程稳定，无明显过拟合。可视化分析进一步验证了其在实际复杂场景中的实用性与鲁棒性。

本实验成果为后续的机器人路径规划与硬件部署提供了坚实的技术基础，也为目标检测模型在体育智能化应用场景中的推广与落地提供了良好的实践范式。

6.2.展望

尽管本实验取得了较为优异的检测效果，但仍存在进一步提升空间。后续工作可从以下几个方向展开：

模型优化升级：进一步探索适用于小目标检测的轻量级结构优化方案，如引入注意力机制（如 SE、CBAM 模块）或结合 Transformer 特征提取模块，以增强模型对复杂特征的表达能力。同时，可考虑结合模型蒸馏技术，将大型模型的检测能力迁移至轻量模型，提升在边缘设备上的性能表现。

部署与系统集成：下一阶段将模型部署至飞腾派等边缘计算平台，并与机器人路径规划模块（如 Dijkstra 算法）进行系统级集成，形成完整的“识别—决策—执行”闭环流程，实现真正意义上的全自动网球捡拾功能。

任务拓展与迁移：在现有单类别检测的基础上，进一步扩展至多类别球类识别任务（如羽毛球、乒乓球等），并探索模型在不同运动场景中的迁移能力，推动其在体育智能化设备中的多样化应用。