基于改进的小孔成像与YOLOv11的锁孔位置及角度识别方法

（一）赛题难点分析及对应解决办法

高压电箱维护需频繁开锁操作，传统人工方式效率低且存在触电风 险，智能识别技术可提升作业安全性与效率。锁孔形状较为复杂、角度偏移，视角俯仰都有可能导致识别困难， 同时涉及开锁等一系列操作，对识别的精度要求高。 赛题需要对锁孔的位置、是否归位、偏转角度以及距离进行检测，算法需要保证一定的实时性和准确性。

针对上述问题，团队提出的方法和工作如下：

1. 提出了一套完整的锁孔定位、锁孔角度检测及锁孔距离测量方案；

2. 针对锁孔角度测量存在输入特征图小、特征不明显、角度难以估计的问题，团队提出了基于HSV的图像增强预处理策略通过对亮度通道增强，增强特征，同时利用obb旋转角度检测网络进行角度检测；

3. 针对相机内参外参缺失的问题，数据测量存在偏差的问题，提出了基于多项式拟合的小孔成像计算方法，相较于传统的小孔成像测量算法，加入了多项式因子，减少计算误差；

（二）数据分析及预处理策略

比赛中给的数据非常少，总共只有 30 张图片。为了准确分析赛题难点，首先对检测物体的尺度进行了分析，比赛数据中，图像中各类级别物体占比如下图所示：

整体来看，box 的分布比较均匀，而锁孔的距离大多集中在 5-25,30 以上的图片相对较少；数据中没有太多小目标数据集；

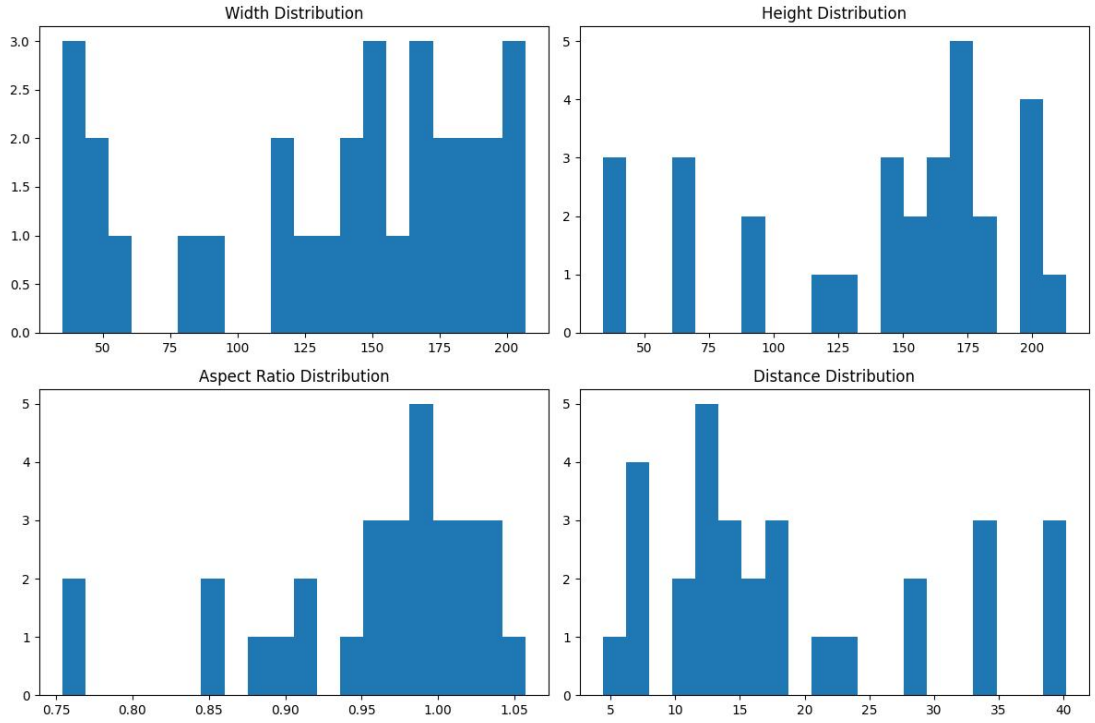


图 1 数据集统计信息

1. 数据增强策略

团队从多个方面进行数据增强

1.扩充数据集：由于数据较少，团队在从互联网上爬取了部分锁孔数据集，通过人工筛选，用于扩充训练数据；部分图像如下图所示：

图 2：扩充数据集

2.数据预处理：利用 albumention 库和 yolov11 内置的数据增强策略，对图像进行翻转裁剪以及 hsv 的通道增强，提高特征提取能力；

图 3 mixup 数据增强

3.训练数据增强：在训练过程中，使用 mixup 算法，对输入的 batch 进行拼接裁剪，提高模型的特征提取能力；如下图所示：

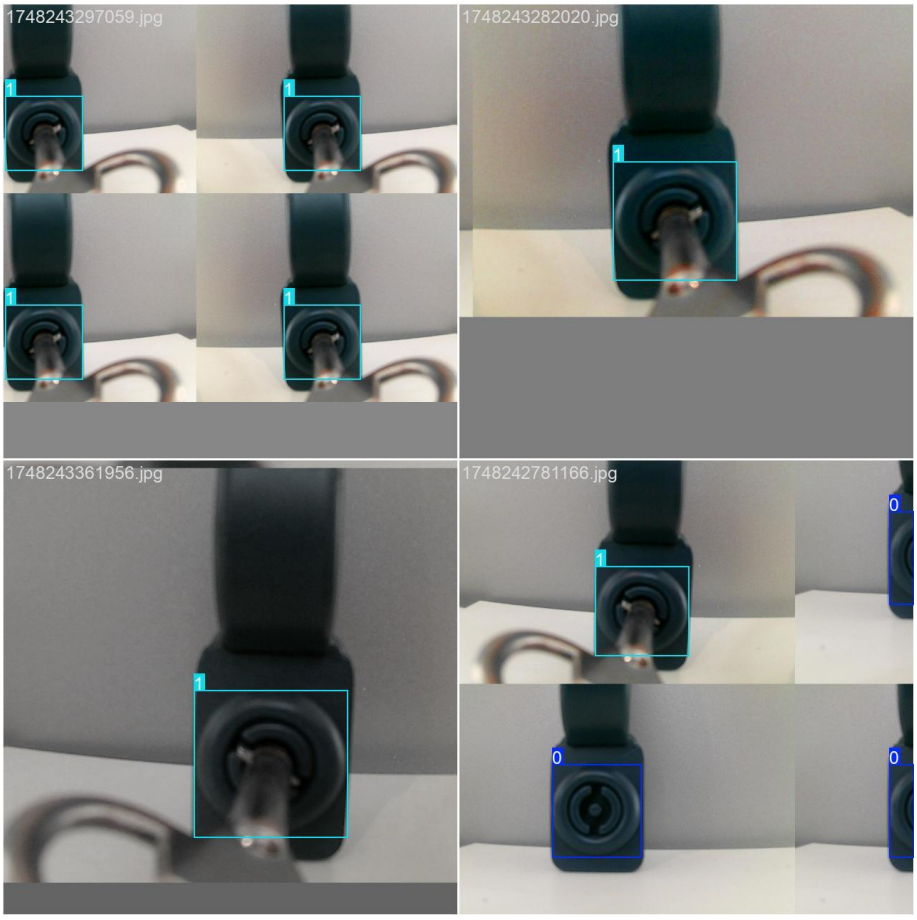


图 4：拼接裁剪增强训练集

（三）算法模型选择

团队将整个方案的实现拆分为三个流程：

1. 锁孔的位置检测及有无钥匙分类；

2. 锁孔的距离检测；

3. 锁孔的旋转角度检测；

下面分别介绍三部分算法的方案选择。

1. 锁孔位置检测算法

考虑到赛题对算法的实时性、准确性以及易实施性都有较高的要

求，团队首先对当前主流的检测算法进行了对比评估，评估结果如下表所示：

名称	检测速度	检测精度	模型大小	部署难度	适用场景	社区支持	备注
YOLOv5	快	中等偏高	中等	低	实时检测、嵌入式设备	高（PyTorch）	轻量级，适合快速部署；支持多平台（ONNX/TensorRT）
YOLOv8	快	高	中等	低	实时检测、移动端	高（PyTorch）	改进版 YOLOv5，精度与速度平衡；官方提供预训练模型
YOLOv11	极快	高	中等	低	实时检测、硬件加速（GPU/FPGA）	高（PyTorch）	最新版本，优化了特征提取模块；支持多尺度检测
YOLOv11x	快	极高	较大	中等	精度优先场景(如工	高（PyTorch）	YOLOv11 的扩展版本，参数

				业质检）		量增加，精度提升显著
SSD	快	中等	小	低	移动端、轻量级场景	中（Caffe/TensorFlow）	单次检测框架，适合简单目标检测
Faster R-CNN	慢	极高	大	高	高精度场景（如医学影像）	高（PyTorch/TensorFlow）	两阶段检测，精度高但计算量大
RetinaNet	中等	高	中等	中等	中等精度需求场景	中（PyTorch）	Focal Loss 解决类别不平衡问题
EfficientDet	中等	高	中等	中等	平衡精度与效率的场景	高（TensorFlow）	自动缩放模型，支持多尺度输入

YOLO 算法的优点在于检测速度快，检测效果好，容易部署在硬件平台上，且第三方资料支持多，便于工程应用实践，考虑到赛题的需求以及比赛时间，团队选择最新的 yolov 11 算法，采用 yolov11 x 模型进行推理测试；

2. 测距算法设计

赛题中除了原始距离并没有给出其他信息，哪怕是摄像头的信息

也只给出部分信息，加上数据集的限制，因此采用较复杂的方案测距并不现实；

为此，研究考虑采用小孔成像算法来计算门锁的距离

小孔成像算法原理如下：

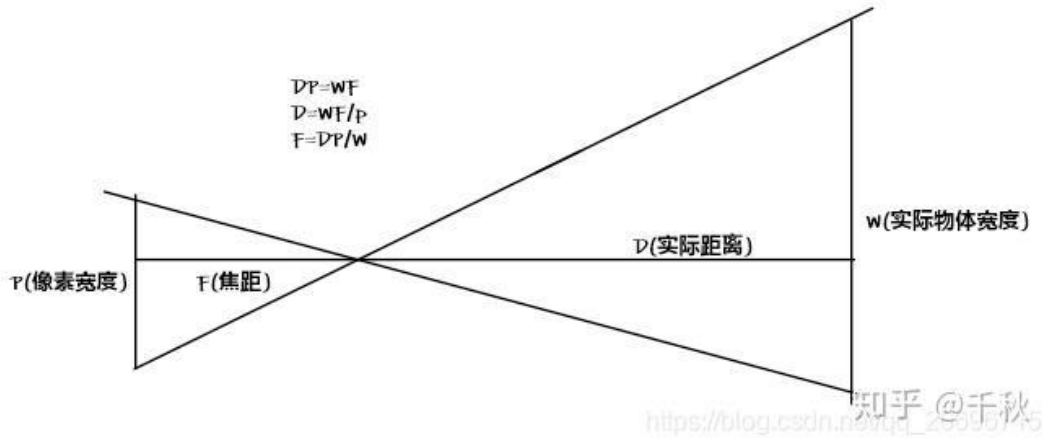


图 5 小孔成像算法原理

假设有一个宽度为 W 的目标或者物体。然后我们将这个目标放在距离我们的相机为 D 的位置。我们用相机对物体进行拍照并且测量物体的像素宽度 P 。这样我们就得出了相机焦距的公式：

$$F = (P \times D) / W$$

举个例子，假设在离相机距离 D = 24 英寸的地方放一张标准的 8.5 x 11 英寸的 A4 纸（横着放；W = 11）并且拍下一张照片。测量出照片中 A4 纸的像素宽度为 P = 249 像素。

因此焦距 F 是：

$$F = (248px \times 24in) / 11in = 543.45$$

当相机移动靠近或者离远物体或者目标时，可以用相似三角形来

计算出物体离相机的距离：

$$D' = (W \times F) / P$$

为了更具体，我们再举个例子，假设我将相机移到距离目标 3 英尺（或者说 36 英寸）的地方并且拍下上述的 A4 纸。通过自动的图形处理我可以获得图片中 A4 纸的像素距离为 170 像素。将这个代入公式得：

$$D' = (11in \times 543.45) / 170 = 35 \text{ 英寸}$$

或者约 36 英寸，合 3 英尺。

从以上的解释中，我们可以看到，要想得到距离，我们就要知道摄像头的焦距和目标物体的尺寸大小，这两个已知条件根据公式：

$$D' = (W \times F) / P$$

得出目标到摄像机的距离D，其中P是指像素距离，W是A 4纸的宽度，F是摄像机焦距。

由于测量误差以及参数不完整带来的影响，g哦单纯用小孔成像的计算公式的结果误差非常大，因此团队加入了一个多项式因子，利用一个3阶的多项式，拟合已有距离的分布，具体公式如下：

$$D' = (W \times F) / P * f(P)$$

$$F(P) = a*P^3+b*P^2+c*P+d$$

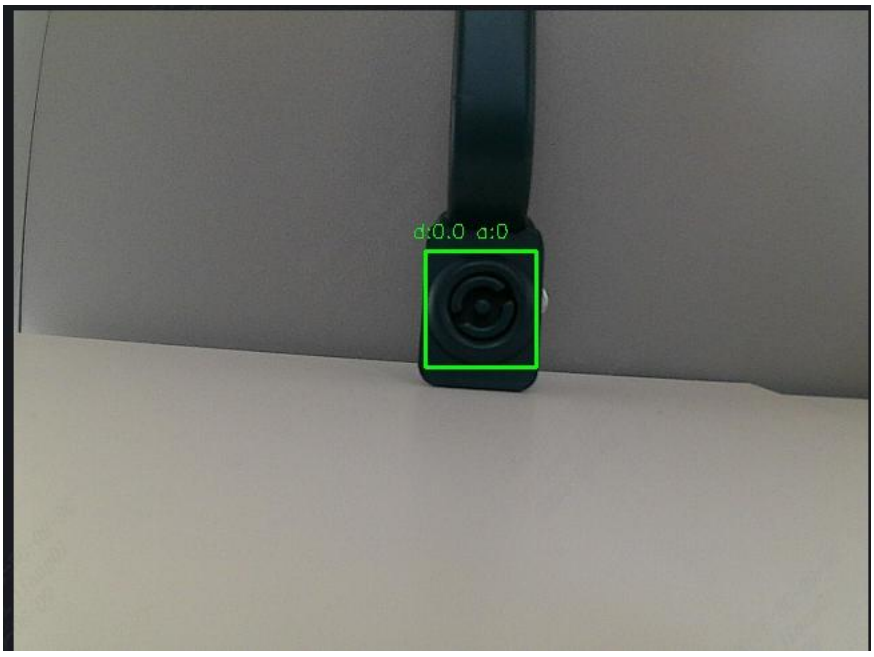
在实际应用中，W 和 F 都是已知值，变量为 P，通过多项式因子，减少了 P 带来的误差。

3．角度测量算法设计

锁孔的角度是否归位以及偏转角度的测量，主要有两种思路，一种利用传统的图像算法，另外一种利用神经网络训练拟合；

①方案 1：利用边缘检测定位轮廓

利用 Canny 算子获取轮廓，并根据轮廓计算锁孔附近开口处，然而，这类算法需要的超参数较多，例如 Canny 算子的边缘阈值，霍夫曼检测直线的曲度，以及对小边缘的过滤。经过图像增强等多种尝试，效果还是不理想，且不稳定，具体如下图所示，团队还是决定采用深度学习的算法。



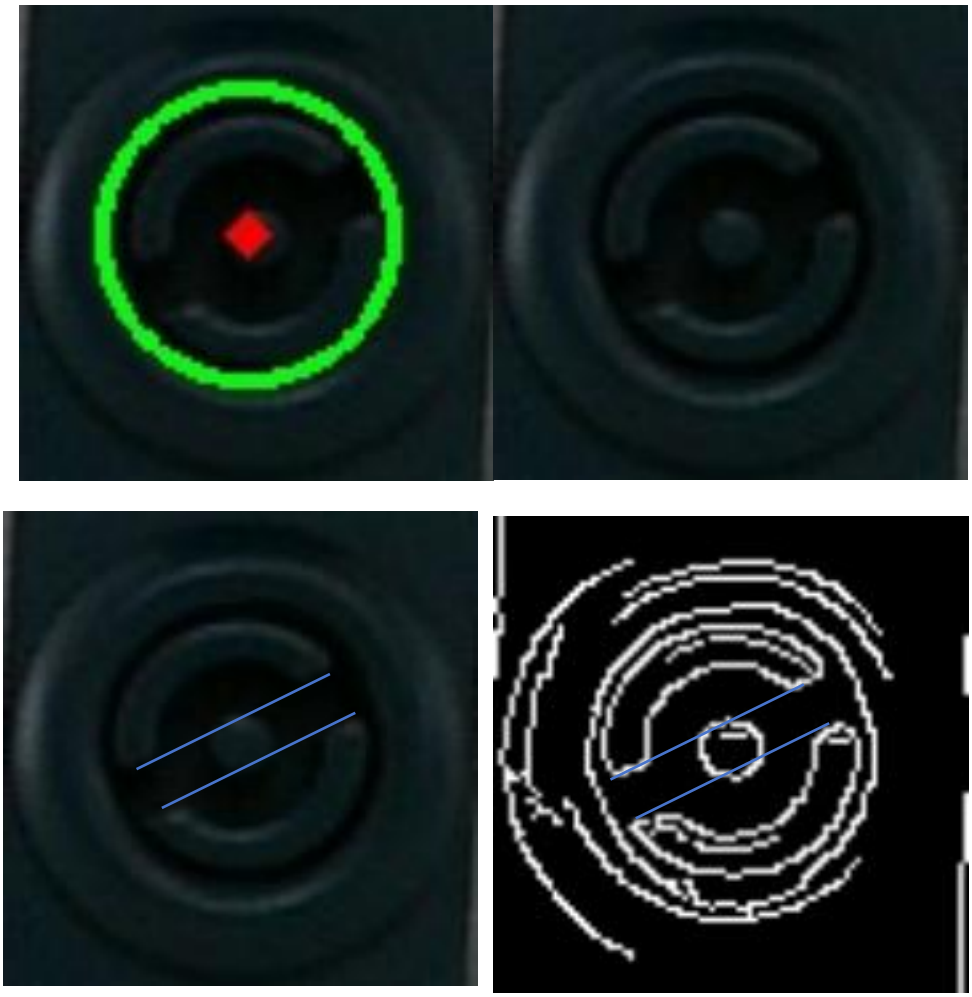


图 6 利用边缘检测定位轮廓检测锁孔角度

（2）方案 2：基于 YOLO 的 OBB 旋转检测

当前基于深度学习的算法已经能够对旋转的目标框进行检测了，然而比赛并没有给出 obb 的旋转检测框，为获取标注框，团队利用 rolabelimg 软件手动标注 OBB 检测框，利用标注文件，对锁孔位置进行 crop,并选择锁孔中间的开口矩形区域作为检测目标；

如果直接用 crop 出来的图片进行检测，检测效果可能一般，因此，算法在输入旋转检测前，对图像进行 HSV 图像增强，并对图像进

行 resize,增强后效果如下图所示：

图 7 增强前后对比

（四）整体方案设计

1. 首先，图片经过目标检测模型，定位锁孔的位置以及对是否存在钥匙进行分类；
2. 检测后的输入测距算法，分析检测框在图像中的占比，并根据预先计算出的相似比，推算锁孔距离在真实世界中的距离；
3. 判断锁孔距离是否大于 30±3cm,如果大于，则不分析锁孔角度，执行第五步，若小于，则执行第四步；
4. 将裁剪出来的锁孔图像输入进行增强，再进行锁孔角度检测；
5. 流程结束；

整体算法架构设计如下图所示：

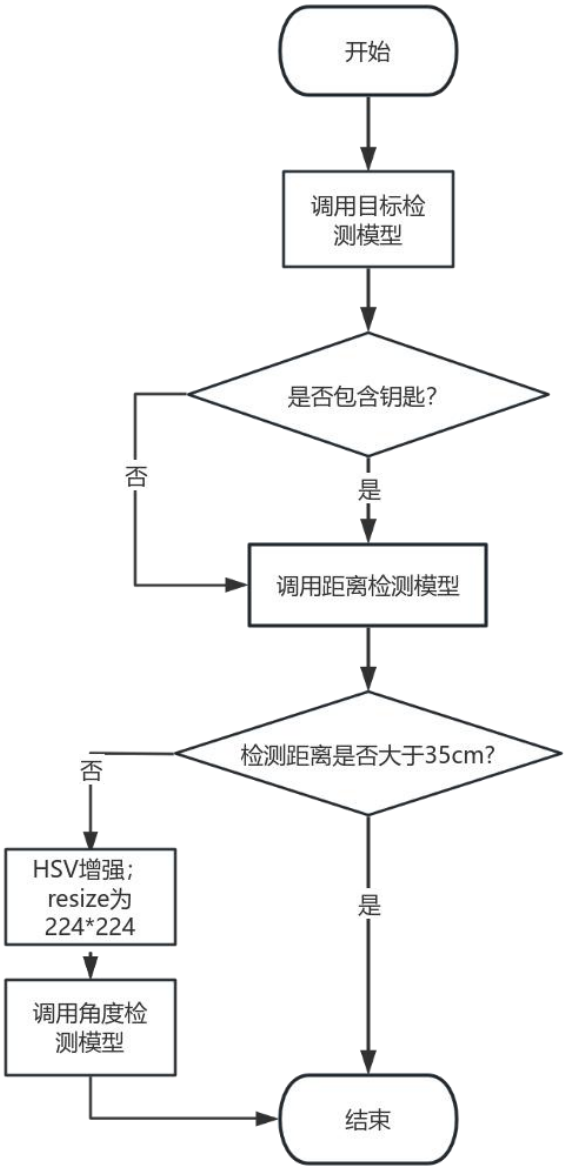


图 8 算法流程图

（五）实验设计及结果分析

1．实验环境

实验环境如下：团队采用的硬件环境为：CUDA:0 (NVIDIA A30, 24062MiB)

软件环境为：20.04.1-Ubuntu，python 3.11.11，具体采用的 python

依赖库如下：

ulturalytics==8.3.149

opencv-python==4.11.0.86

numpy==2.2.4

onnx

torch

2．实验方法：

为评估不同超参数、数据增强策略以及测距参数对算法效果的影响，研究设计以下实验进行评估:

1. 检测算法的收敛速度，以及不同模型的检测速度对比；
2. 不同乘法因子对测距误差的影响；
3. 角度检测算法的收敛速度分析
4. 数据增强对角度检测算法的影响评估

3．目标检测算法实验设计及结果分析

超参数对目标检测算法预训练结果影响，在测试集的训练结果如下：

	precision(B)	recall(B)	mAP50(B)	mAP50-95(B)
Epoch = 100	0.74407	0.5	0.745	0.52619
Epoch	0.79059	0.83333	0.89833	0.62866

=200				
Epoch= 300	0.83642	1	0.94786	0.67817
Epoch = 500	0.98302	1	0.995	0.7197

Epoch=500 时，模型的 box loss 和 cls loss 逐渐收敛，模型的 box loss 变化趋势为例，如下图所示：

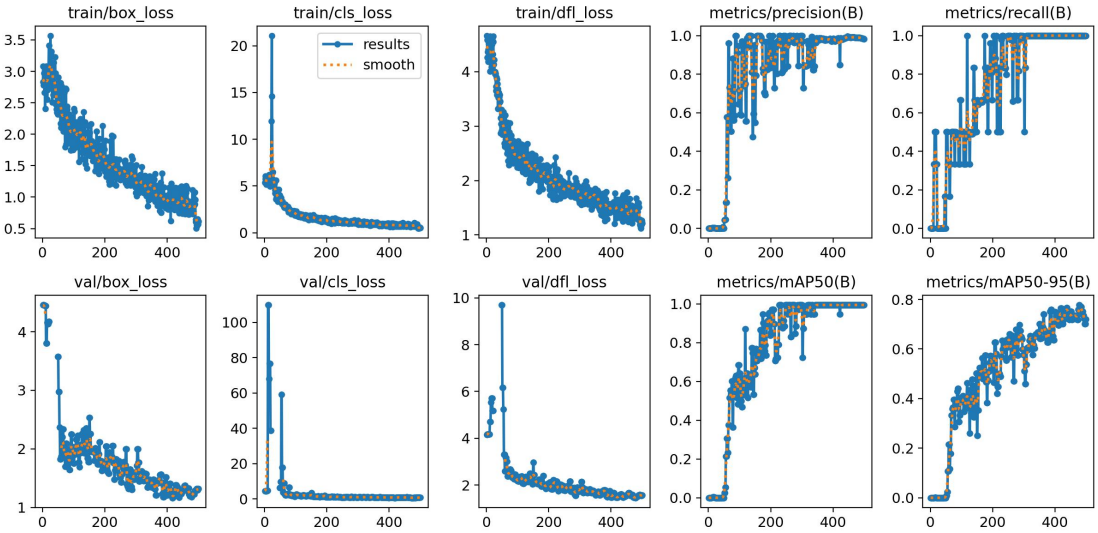


图 9 检测算法损失变化趋势

从图中可以发现，大多数损失在 epoch=400 之后开始慢慢收敛，box 的 loss 收敛较慢，其中部分原因来自数据集太少。

4. 改进后的小孔测距算法实验设计及结果分析

使用简单的小孔成像算法拟合距离误差非常大，团队对比了二次函数、一次函数、多项式拟合、反比例函数对测距结果的影响，误差分布如下表所示：

	最大误差 (mm)	最小误差 (mm)	平均误差 (mm)
多项式因子	3.63	0.02	1.13
反比例函数	7.39	0.04	2.33
二次函数	8.21	0.15	2.06
线性函数	8.69	0.07	3.06
Baseline	396.58	67.45	135.65

从表格来看，当目标距离越远，检测的距离误差越大，因此，为了更好拟合距离分布，采用多项式因子来拟合。

5. 锁孔角度检测实验设计及结果分析

由于 crop 出来的锁孔 box 已经非常小，人眼有非常难辨别锁孔的位置，且锁孔与锁的周围颜色分布类似，因此研究设计了 resize+hsv 增强图像用于旋转目标检测，增强后对比效果如下：

图 10 HSV 通道图像增强与 resize

检测精度和损失变化如表所示：

	precision(B)	recall(B)	mAP50(B)	mAP50-95(B)
Epoch =	0.74407	0.5	0.745	0.52619

100				
Epoch =200	0.79059	0.83333	0.89833	0.62866
Epoch=300	0.83642	1	0.94786	0.67817
Epoch = 500	0.98302	1	0.995	0.7197

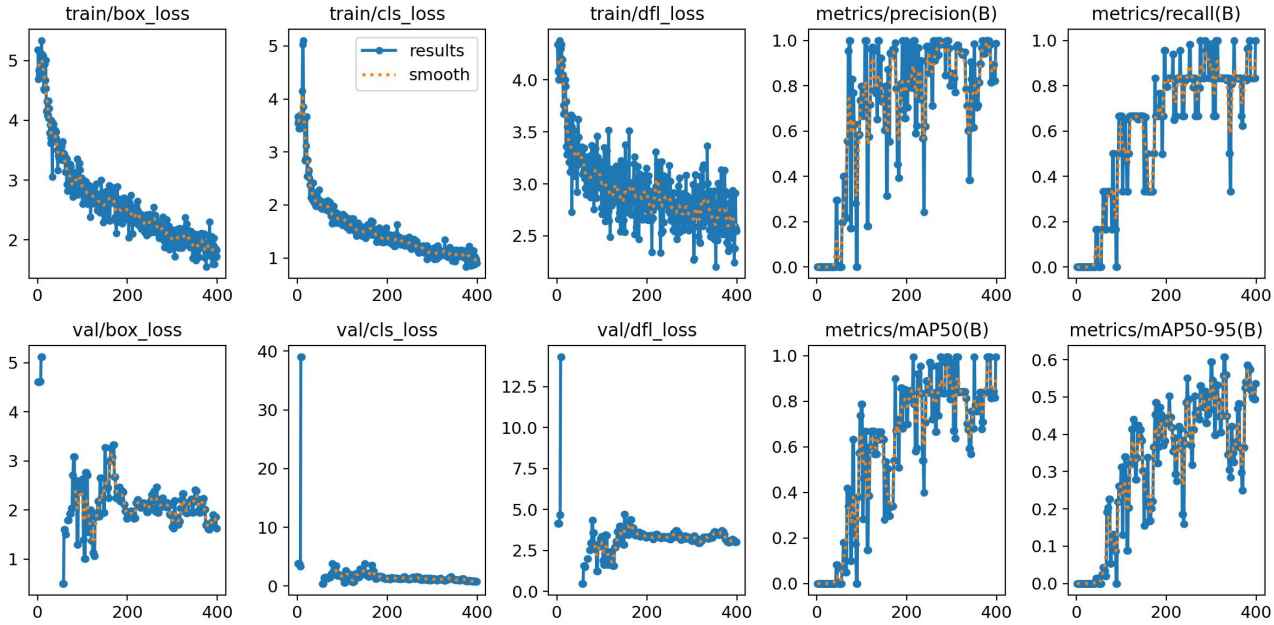


图 10 OBB 角度检测网络损变化趋势

（六）总结

具体来说，团队的工作主要如下：

1. 提出了一套完整的锁孔定位、锁孔角度检测及锁孔距离测量方案；

2. 针对锁孔角度测量存在输入特征图小、特征不明显、角度难以估计的问题,团队提出了基于 HSV 的图像增强预处理策略通过对亮度通道增强,增强特征，同时利用 obb 旋转角度检测网络进行角度检测；

3. 针对相机内参外参缺失的问题，数据测量存在偏差的问题，提出了基于多项式拟合的小孔成像计算方法，相较于传统的小孔成像测量算法，加入了多项式因子，减少计算误差；