

2026赛季视觉组选拔考核综合技术报告

任务二：模型训练 & 附加任务：曝光优化

考核人：[你的名字]

2026 年 1 月 15 日

目录

1 考核任务二：YOLO 模型训练实战与思考	2
1.1 项目概述	2
1.2 实战流程	2
1.2.1 1. 数据集准备 (Data Preparation)	2
1.2.2 2. 数据标注 (Labeling)	2
1.2.3 3. 数据集划分 (Dataset Splitting)	2
1.2.4 4. 模型训练 (Model Training)	2
1.3 思考题：魔方角点识别	3
1.3.1 Q1: 为什么我们需要角点？(Why?)	3
1.3.2 Q2: 如何准确识别角点？(How?)	3
2 附加考核：复杂光照下的曝光优化调研	5
2.1 问题背景	5
2.2 现有算法分析	5
2.3 解决方案调研	5
2.3.1 方案 A: ROI 自动曝光 (硬件层优化) - 推荐	5
2.3.2 方案 B: Gamma 校正 (图像预处理)	5
2.3.3 方案 C: 自适应直方图均衡化 (CLAHE)	6
2.4 演示系统实现	6

1 考核任务二：YOLO 模型训练实战与思考

1.1 项目概述

本项目基于 YOLOv11 框架，完成了针对赛场关键元素（台球 ball、魔方 cube）的目标检测模型训练。项目涵盖了从视频数据清洗、数据集构建、模型训练到推理验证的全流程。

1.2 实战流程

1.2.1 1. 数据集准备 (Data Preparation)

使用 Python 脚本从原始视频（raw.mp4, raw1.mp4, raw2.mp4）中提取训练图像。为了保证数据集的多样性并防止文件覆盖，编写了 cut_final.py 脚本，支持自定义采样步长和文件名前缀。

```
1 # Core Logic: Save frame by step and rename
2 if current_frame % step == 0:
3     filename = f"{output_folder}/{image_prefix}_{saved_count}.jpg"
4     cv2.imwrite(filename, frame)
5     saved_count += 1
```

Listing 1: 视频抽帧核心代码 (cut_final.py)

1.2.2 2. 数据标注 (Labeling)

使用 X-AnyLabeling 工具进行标注，采用 YOLO 格式导出。

- 类别定义：

- Class 0: ball (台球)
- Class 1: cube (魔方)

- 标注策略：手动剔除模糊图像，确保每个目标均有独立包围框 (Bounding Box)。

1.2.3 3. 数据集划分 (Dataset Splitting)

编写 split_data.py 脚本，自动将图像和对应的标签文件按 8:2 的比例随机划分为训练集 (train) 和验证集 (val)，并整理为 YOLO 标准目录结构。

1.2.4 4. 模型训练 (Model Training)

基于 yolo11n.pt 预训练模型进行迁移学习。训练参数配置如下：

- Epochs: 50

- Batch Size: 8
- Image Size: 640
- Device: GPU (CUDA)

```

1 from ultralytics import YOLO
2
3 if __name__ == '__main__':
4     model = YOLO('yolo1in.pt')
5     model.train(
6         data='data.yaml',
7         epochs=50,
8         imgsz=640,
9         batch=8,
10        workers=0, # Must be 0 on Windows
11        device='0',
12        name='task2_result'
13    )

```

Listing 2: 训练启动脚本 (train.py)

1.3 思考题：魔方角点识别

1.3.1 Q1: 为什么我们需要角点? (Why?)

目前的 YOLO 模型仅能输出 2D 边界框 (Bounding Box)，这在机械臂抓取任务中存在局限性：

1. **PnP 解算需求：** PnP (Perspective-n-Point) 算法需要利用物体在世界坐标系中的 3D 点与图像中的 2D 像素点进行匹配，从而求解相机的位姿（旋转矩阵 R 和平移向量 t ）。角点是进行点对点匹配的最佳特征。
2. **姿态估计：** 仅凭矩形框无法获知魔方的旋转角度（如平放、斜放）。通过识别角点，可以精确计算魔方的 6DoF 姿态，辅助机械臂规划抓取路径。

1.3.2 Q2: 如何准确识别角点? (How?)

针对魔方角点识别，提出以下两种技术路线：

方案 A：基于深度学习的关键点检测 (Keypoint Detection)

- **思路：** 利用 YOLO-Pose 等模型，将魔方的 8 个角点定义为“骨骼关键点”进行训练。
- **优点：** 端到端输出，鲁棒性强，受光照和背景干扰较小。

- 缺点：标注成本极高，需要对每个角点进行精确打点。

方案 B：传统视觉 + 几何约束 (Traditional CV)

- 思路：YOLO 裁剪 ROI区域 → Canny 边缘检测 → 霍夫变换 (Hough Lines) 拟合直线 → 计算直线交点。
- 优点：不需要大量训练数据，推理速度快。
- 缺点：对边缘磨损和复杂光照敏感，容易产生误检。

2 附加考核：复杂光照下的曝光优化调研

2.1 问题背景

在实际比赛中（尤其是高纬度地区赛场），常出现极端的高反差（High Contrast）光照条件。由于地面反光强烈，相机自动曝光（AE）算法受全图平均亮度误导，降低了曝光时间，导致位于阴影中的暗色目标（黑桶）细节完全丢失。

2.2 现有算法分析

传统的曝光检测算法通常基于全局亮度均值或全局直方图统计：

- **缺陷：**当画面中存在大面积高亮区域（如反光地面）时，全局均值被拉高，掩盖了局部暗部过暗的事实。
- **验证：**使用编写的 app.py 工具分析去年的比赛视频截图，发现其亮度直方图极不均匀，且暗部区域对比度极低。

2.3 解决方案调研

2.3.1 方案 A：ROI 自动曝光（硬件层优化）- 推荐

原理：通过相机 SDK 接口（如 SetAutoExposureROI），指定相机仅根据画面中央区域（通常是目标出现的位置）进行测光，忽略周围高亮地面的影响。**优势：**从物理层面增加感光元件的积分时间，获取更多暗部光子信息，信噪比最高。

2.3.2 方案 B：Gamma 校正（图像预处理）

原理：利用非线性变换提升暗部细节，同时抑制亮部过曝。公式如下：

$$V_{out} = V_{in}^{\gamma} \quad (\text{其中 } \gamma < 1) \quad (1)$$

当 γ 取 0.5 左右时，可以显著提亮低灰度区域。

```
1 def apply_gamma_correction(image, gamma=0.5):
2     invGamma = 1.0 / gamma
3     table = np.array([(i / 255.0) ** invGamma) * 255
4                         for i in np.arange(0, 256)]).astype("uint8")
5     return cv2.LUT(image, table)
```

Listing 3: Gamma 校正实现代码

2.3.3 方案 C: 自适应直方图均衡化 (CLAHE)

原理: 不同于全局直方图均衡化 (HE), CLAHE 将图像划分为多个小块 (Tiles) 分别进行均衡化, 并限制对比度阈值以防止噪声放大。 **优势:** 能有效增强局部纹理细节, 非常适合处理“黑桶内部看不清”的问题。

```
1 def apply_clahe(image):
2     lab = cv2.cvtColor(image, cv2.COLOR_BGR2Lab)
3     L, A, B = cv2.split(lab)
4     # ClipLimit limits contrast amplification
5     clahe = cv2.createCLAHE(clipLimit=2.0, tileSize=(8, 8))
6     L = clahe.apply(L)
7     return cv2.cvtColor(cv2.merge([L, A, B]), cv2.COLOR_Lab2BGR)
```

Listing 4: CLAHE 实现代码

2.4 演示系统实现

为了验证上述算法, 开发了基于 Flask 的 Web 演示工具 `app.py`。

- **诊断功能:** 集成亮度直方图、局部对比度分析、连通区域检测等 5 种算法判断图像是否过曝。
- **优化功能:** 实时对比原图、线性调整、HE 以及 CLAHE 的处理效果。
- **结论:** 实验表明, **CLAHE** 算法在恢复暗部纹理方面效果最佳, 建议与 ROI 自动曝光配合使用。