/ /

Week 6 2025.11.3 ~ 2025.11.9

Goal

Stabilize the Ultralytics (YOLO) training / inference environment and standardize dataset / project structure for subsequent det / seg experiments

Done

1. Installed and validated Pytorch (CUDA) + Ultralytics on Windows; conda; confirmed GPU training readiness via a dry-run inference and a minimal training stanity check.
2. Standardized repository layout (datasets / runs / scripts) and experiment naming; enforced fixed seeds and consistent output logging.
3. Organized RDD 2022 (China_Drone, China_MotorBike) for detection and CRACK500 for segmentation; verified paths, labels, and train / val splits; prepared dataset YAMLs.

Issues

1. Notable domain gap between RDD 2022 (wide-view road scenes) and CRACK500 (thin-structure close-view labels), implying scale mismatch risk for direct segmentation on full-frame images.
2. Windows path depth / space issues may break batch scripts; needs strict path conventions.

Plan

Start a baseline RDD2022 det training run (small model, moderate imgsz) and set up evaluation + visualization outputs for fast iteration.

Week 7 2025.11.10 ~ 2025.11.16

Goal

Train a fast baseline detection model on RDD2022 China subsets and verify its ability to localize crack candidate regions in real-road imagery.

Done

1. Trained a baseline YOLO detection model on RDD2022 (China_Drone, China_MotorBike) with reproducible configs (seeds, epochs, imgsz).
2. Verified inference on held-out samples; exported visual overlays to check whether boxes align with crack-like regions under varying textures/illumination.
3. Logged baseline metrics (mAP/precision/recall) and collected typical failure cases (glare, lane markings, transverse joints).

Issues

1. False positives under strong texture patterns and markings; confidence/NMS thresholds heavily affect perceived quality.
2. Some annotations are ambiguous or incomplete and small scale, which can cap achievable precision.

Plan

Tune thresholds and augmentations; run a second det experiment with controlled changes (one-variable-at-a-time) and compare failure modes.

/ /

Week 8 2025.11.17 ~ 2025.11.23

Goal

Improve detection robustness and produce a "useble" det checkpoint for downstream cascade (det -> seg) integration.

Done

1. Ran at least one follow-up det training with adjusted hyper-parameters / augmentations; compared results against the baseline using the same validation split.
2. Conducted error analysis by scene types ( drone vs motorbike viewpoint ) and summarized the main FP / FN patterns.
3. Selected a "best-so-far" det weight for cascade experiment and documented its intended operating thresholds.

Issues

1. Drone-view scenes tend to produce larger candidate areas; segmentation may collapse if ROI is too large or overly downscaled later.
2. Over-aggressive augmentation can increase recall but worsen FP density in textured asphalt.

Plan

Start preparing CRACK500 segmentation training (YOLO-seg format + patch strategy and define a consistent post-processing rule for masks.

/ /

Week 9 2025. 11. 24 ~ 2025. 11. 30

Goal

Build the segmentation pipeline baseline using CRACK500 and ensure labels | training data match "thin crack" characteristics.

Done

1. Converted / validated CRACK500 segmentation labels and dataloader; ensured masks align with images after any resizing | cropping.
2. Trained a baseline YOLO segmentation model (lightweight) and checked qualitative outputs on validation patches.
3. Implemented basic mask post-processing trials (binarization + simple morphology) to reduce speckle noise.

Issues

1. Class imbalance (thin positive pixels vs large background) can cause unstable masks early in training.
2. Patch sampling must preserve crack continuity; random crops can cut long cracks and bias learning.

Plan

Refine patch generation and augmentation policy, re-train seg with a more stable configuration and compare mask connectivity.

/ /

Week 10 2025.12.1 ~ 2025.12.7

Goal

Push segmentation quality from "works" to "useable": sharper thin structures, fewer broken segments, and controllable noise.

Done

1. Re-trained seg with improved settings (imgsz / augmentation / epochs) focusing on thin-structure preservation and mask continuity.
2. Evaluated qualitative mask failure cases (fragmentation, over-thickening, texture-induced noise) and adjusted post-processing accordingly.
3. Saved a candidate seg checkpoint for cascade integration and wrote a minimal inference script for patch-level testing.

Issues

1. Segmentation can overfit to CRACK500 texture statistics, reducing generalization to wide-view RDD-style scenes.
2. Morphology can repair fragmentation but may distort true crack width; needs conservative parameters.

Plan

Implement cascade inference Det → ROI crop / scale → Seg → paste back, starting from single-image end-to-end verification.

/ /

Week 11 2025.12.8 ~ 2025.12.14

Goal

Implement the first working cascade inference prototype and confirm correct coordinate mapping (ROI ↔ full image).

Done

1. Implemented cascade v1: dection → extract ROI → resize to seg input → run segmentation → map mask back to original coordinates.
2. Produced visual outputs (overlay with boxs + masks) for single-image tests to validate correctness and debug geometry.
3. Added basic filtering rules (min box size / confidence threshold) to avoid running seg on obvious negatives.

Issues

1. Large ROIs lead to downscaling; thin cracks may vanish → "empty mask" risk.
2. Boundary artifacts appear when pasting masks back if resizing / interpolation is inconsistent.

Plan

Add adaptive ROI handing (dynamic scaling; optional tilling for oversized ROIs) and test on more diverse samples.

/ /

Week 12 2025. 12.15 ~ 2025. 12.21

Goal

Make cascade stable under scale variance by introducing tilling + overlap + stitching for oversized ROIs.

Done

1. Implemented ROI tilling for large boxes (sliding window with overlap) and stitched tile masks back into a single ROI mask.
2. Added debug outputs for each stage (ROI crops, title grids, stitched masks) to verify coverage and reduce seam artifacts.
3. Verified cascade behavior on cases with very large detected regions (e.g., road blocks/markings) and adjusted size thresholds

Issues

1. Stitching seams and overlap handling can create discontinuities; merging rule must be consistent (e.g., max/OR).
2. Tiling increases compute; needs a cap strategy to avoid worst-case latency.

Plan

Extend from single-image to batch folder inference, standardize output naming, and record run metadata for reproducibility.

/ /

Week 13 2025.12.22 ~ 2025.12.28

Goal

Operationalize the pipeline: batch processing, structured outputs, and repeatable run records.

Done

1. Implemented batch inference for image folders; automatic per-image outputs (mask PNG + overlay visualization) with deterministic naming.
2. Added run metadata logging (weights used, thresholds, imgsz, dataset tag) to a single run info file for traceability.
3. Performed small-scale batch tests and summarized common failure cases for both det and seg stages.

Issues

1. Failure cases cluster by domain; glare / texture affects det; scale context mismatch affects seg despite ROI fixes.
2. Output directory depth and long experiment names can still cause path issues on Windows

Plan

Consolidate a "baseline end-to-end" run with fixed parameters and prepare a concise mid-term-ready summary (what works, what fails, why).

/ /

Week 14 2025.12.29 ~ 2026.1.4

Goal

Freeze a stable baseline system (det + seg + cascade) and document the "current best" settings and limitations.

Done

1. Ran end-to-end evaluation on representative samples and confirmed the complete workflow: det localization -> seg refinement -> final visualization outputs.
2. Finalized the baseline operating parameters (confidence / NMS, ROI size thresholds, tiling overlap) and recorded them as the current reference configuration.
3. Summarized next-step priorities: generalization checks, cross-dataset validation, and potential model upgrades (YOLOv11 comparision / lightweight enhancements.

Issues

1. Generalization remains the main risk: segmentation trained on CRACK500 may still miss cracks in wide-view scenes without careful ROI strategy.
2. Precision/recall trade-offs depend strongly on thresholds; needs a principled tuning protocol on a small local validation set.

Plan

Begin structured experiments for model comparison (YOLOv8 vs YOLOv11) and ablations; expand to video inference once batch image pipeline is stable.