Sketch-Vision: Primitive Detection and Program Reconstruction

D1.3 Update Report

Team JAXAXAX

Nikita Zagainov (n.zagainov@innopolis.university)
Nikita Tsukanov (n.tsukanov@innopolis.university)
Tetkin Dmitry (d.tetkin@innopolis.university)

November 24, 2025

Abstract

This update documents repository changes since the previous interim report: expanded primitive types (arrows, text tokens), a simple program serialization within annotations, OCR utilities, sequence-level evaluation, and a minimal detector training baseline. We also outline near-term improvements and longer-term extensions.

1 Repository

https://github.com/V1adych/sketch-vision-pmldl

2 Overview

We extend raster-to-program modeling to hand-drawn sketches with engineering annotations. From an input image, we produce a set of detected primitives (geometry and annotation tokens) and a compact symbolic "program" suitable for downstream CAD or sequence modeling.

3 Implemented Changes

3.1 Expanded Synthetic Generator

We extended `preprocessing/generate_synthetic.py` with two new primitive types: `arrow` and `text`. Annotations now include:

- a `program` field: serialized sequence of primitives suitable for encoder–decoder evaluation;
- an `ocr_gt` list: ground-truth tokens for text primitives.

3.2 Visualization

`preprocessing/visualize_annotations.py` now renders the added types with distinct colors and labels for quick inspection.

3.3 OCR Utility

`preprocessing/extract_text_tokens.py` optionally integrates Tesseract OCR to extract tokens from images and stores them as `ocr_pred` alongside annotations.

3.4 Metrics and Evaluation

`evaluation/metrics.py` includes precision/recall/F1 and sequence helpers (exact match, char accuracy). `evaluation/sequence_metrics.py` evaluates predicted programs (TSV/CSV: `name\t program`) against ground truth.

3.5 Detector Baseline

`scripts/train_detector.py` provides a minimal Faster R-CNN training loop (torchvision) for the synthetic dataset.

4 Quickstart (Updated)

```
python preprocessing/generate_synthetic.py --output-dir dataset/synthetic --num-samples 100
python preprocessing/visualize_annotations.py \
  --images-dir dataset/synthetic/images \
  --annotations-dir dataset/synthetic/annotations \
  --name sketch_00010 --output dataset/synthetic/vis
python evaluation/evaluate_synthetic.py \
  --annotations-dir dataset/synthetic/annotations \
  --splits dataset/synthetic/splits/train.txt

# OCR (optional; requires Tesseract installed)
python preprocessing/extract_text_tokens.py \
  --images-dir dataset/synthetic/images \
  --annotations-dir dataset/synthetic/annotations

# Train simple detector baseline
python scripts/train_detector.py --data-root dataset/synthetic --epochs 1 --batch-size 2

# Evaluate sequence-level accuracy (TSV: name<TAB>program)
python evaluation/sequence_metrics.py \
  --annotations-dir dataset/synthetic/annotations \
  --splits dataset/synthetic/splits/test.txt \
  --predictions predictions.tsv
```

5 Data Visualization

We include dataset summary plots generated from the synthetic annotations (see figures in `docs/figs/`). The visualization tool overlays bounding boxes and labels directly on images, now including arrows and text tokens.

6 Future Improvements

- **Richer primitives**: hatching, construction lines, dimension arrows with explicit endpoints, symbols.
- **OCR robustness**: trainable OCR head or integration with lightweight OCR tailored to sketch digits and units; post-processing with language priors (e.g., ranges, units, tolerances).
- **Detector baselines**: DETR/Deformable DETR variants; add class-agnostic NMS and better augmentation.

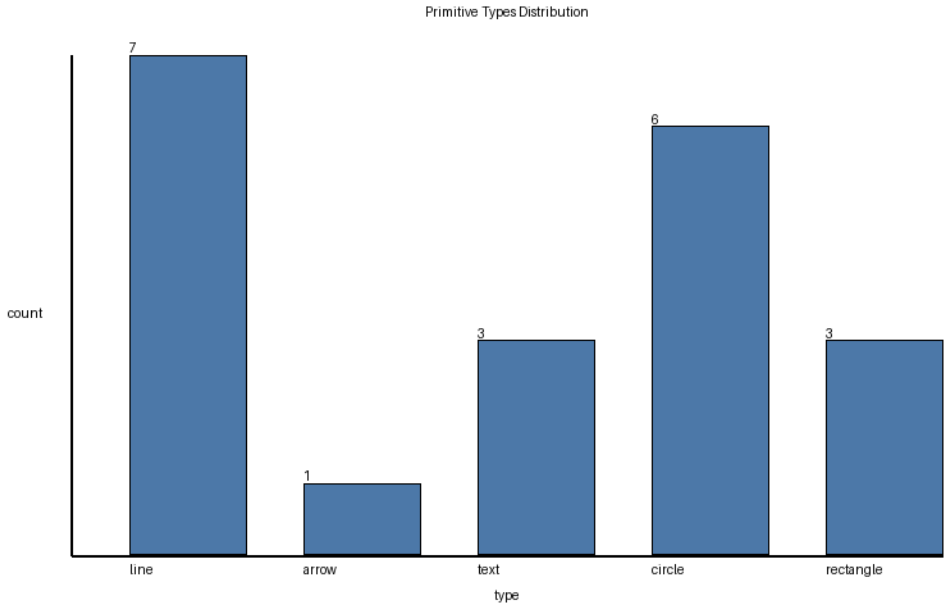

Figure 1: Distribution of primitive types in the dataset. Shows the frequency of each type (rectangle, circle, line, arrow, text) across all annotations.

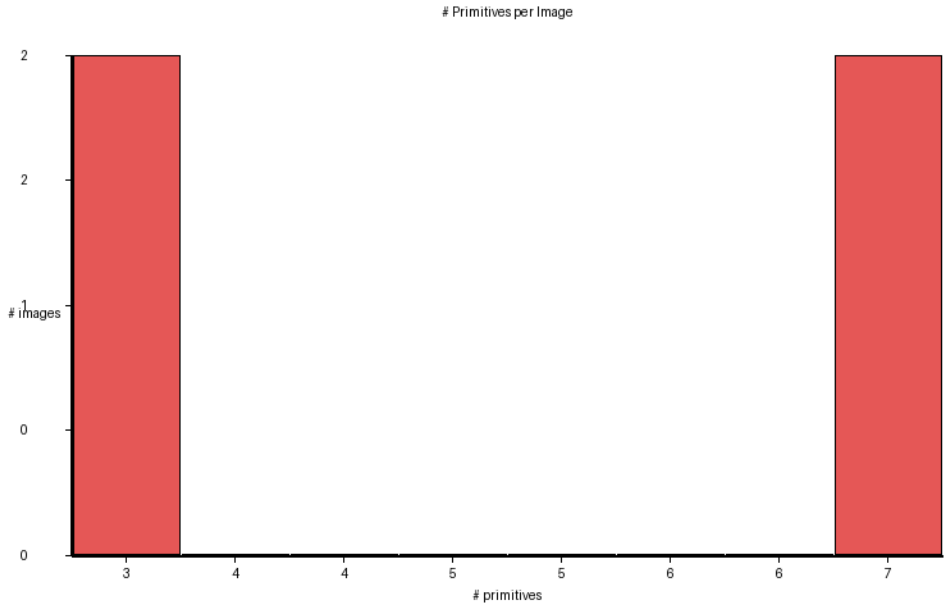

Figure 2: Histogram of primitive counts per image. Most images contain 3–7 primitives, with a typical range of 2–5 primitives per sketch.

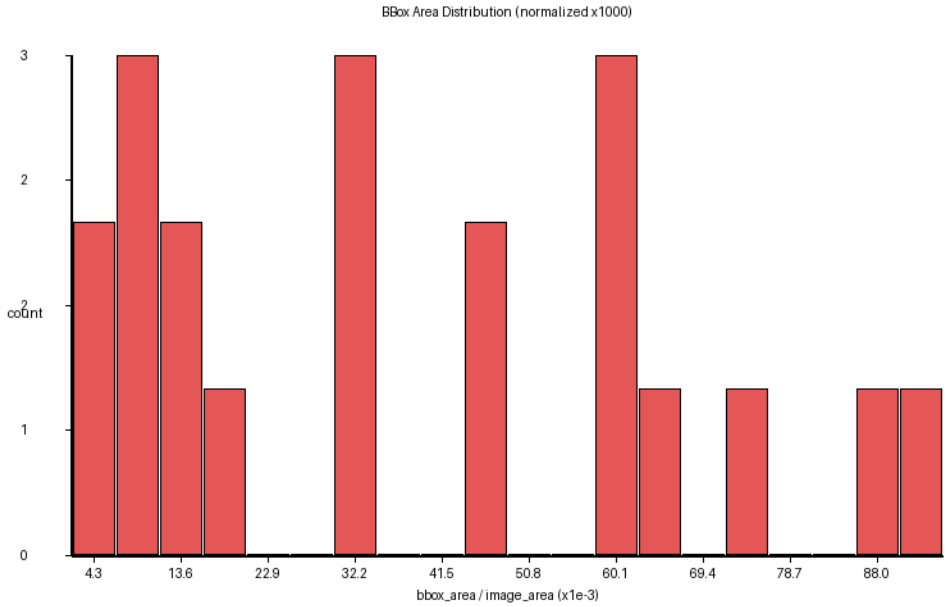

Figure 3: Bounding-box area distribution normalized by image area. Most primitives occupy a small fraction of the image (typically 0.1–5% of total area).

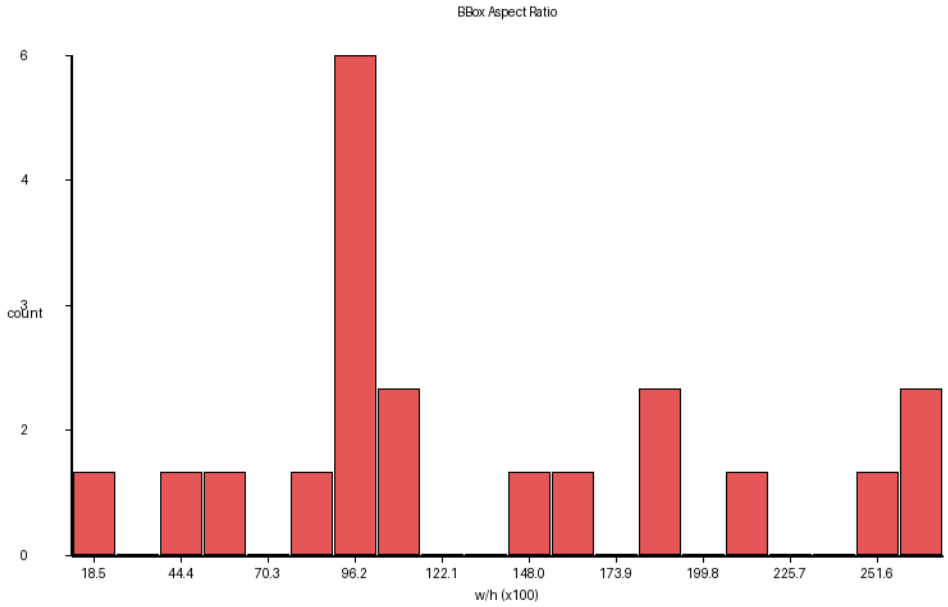

Figure 4: Bounding-box aspect ratio distribution (width/height). Shows the shape diversity of primitives, with values near 1.0 indicating square-like shapes and higher values indicating elongated objects (e.g., arrows, lines).

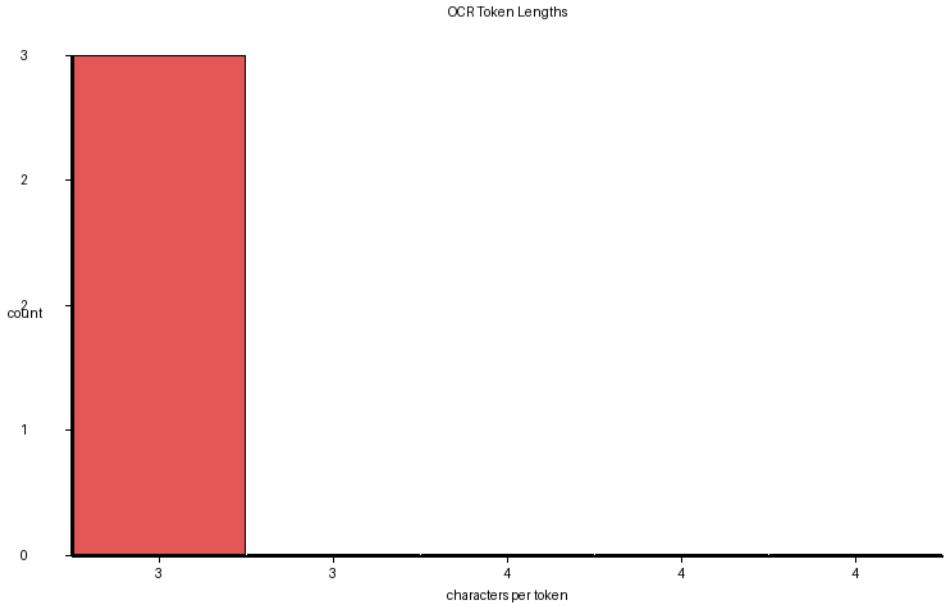

Figure 5: OCR token length distribution (characters per token). Text primitives in the dataset typically contain 1–3 characters, reflecting common dimension labels and numeric values.

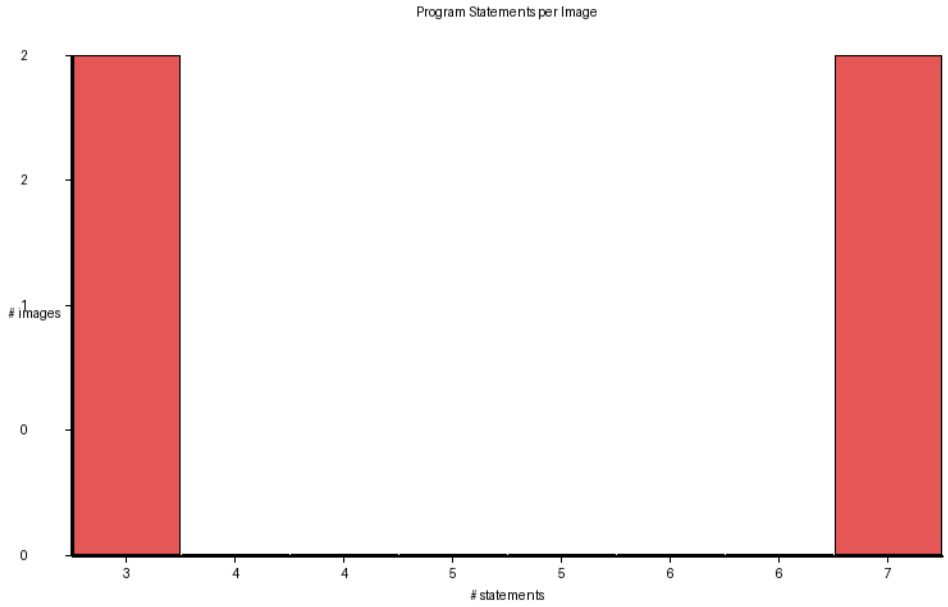

Figure 6: Program statement count per image. The number of statements in the serialized program representation correlates with the number of detected primitives, typically ranging from 2–6 statements per image.

- **Matching and metrics**: Hungarian matching over sets of primitives with type/value constraints; edit-distance metrics for program strings.
- **Encoder–decoder**: integrate a sequence model (ViT encoder + small LM) for end-to-end program prediction and report sequence exact match on synthetic and real sketches.
- **Domain adaptation**: noise models and fine-tuning on photographed hand sketches; semi-supervised learning with pseudo-labels.
- **Tooling**: export to common CAD sketch formats or intermediate DSLs; visualization dashboards.

7 Reproducibility Notes

Dependencies are listed in `requirements.txt`. For detector training, install compatible `torch` and `torchvision`. The repository includes deterministic seeds for synthetic data generation.