CS444 Final PROJECT Report : PERCEPTION CHALLENGE FOR BIN-PICKING BY OPENCV AND INTRINSIC

SAI ROHIT MURALIKRISHNAN
srm17@illinois.edu

HARISH KUMAR BALAJI
hkb2@illinois.edu

VIKRAM RAJ
vn18@illinois.edu

YUDAI YAMADA
yyamada2@illinois.edu

This project presents a model for predicting the 6 Degrees of Freedom (6-DoF) required for robotic grasping using RGB-D imagery and associated camera intrinsics. Our pipeline integrates object detection to localize objects via bounding boxes, followed by a pose prediction module that estimates position and orientation — (X, Y, Z, Roll, Pitch, Yaw) — for each detected object.

1 Introduction

Accurate 6-DoF pose estimation for bin-picking is a fundamental challenge in industrial automation, where robotic systems must detect and localize objects in cluttered scenes for reliable grasping. Occlusion, object similarity, and varied poses make this a non-trivial perception problem, especially under real-time constraints.

Traditional methods using hand-crafted features and point cloud alignment often fail in dense scenes. Modern approaches leverage RGB-D inputs and deep learning to jointly exploit appearance and geometric cues for object detection and pose prediction.

In this project, we address the **OpenCV Bin-Picking Challenge** by developing a two-stage pipeline:

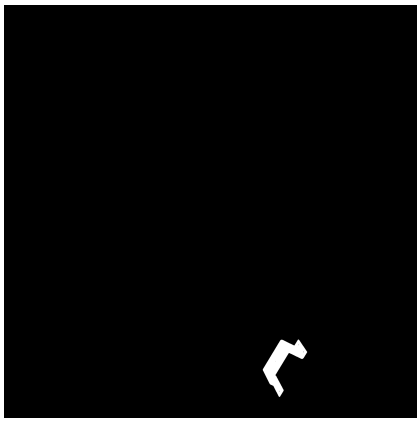
- **Object Detection:** RGB and depth images are processed through a custom detection pipeline to identify object locations.
- **Pose Estimation:** Cropped RGB patches are passed through a **ResNet-18-based network**, which extracts features and regresses the 6-DoF pose via sequential fully connected layers.

We explored several alternative approaches but ultimately chose this method due to computational and integration constraints. Our final model balances performance and efficiency using standard RGB-D sensors and lightweight CNNs for end-to-end 6-DoF pose prediction.

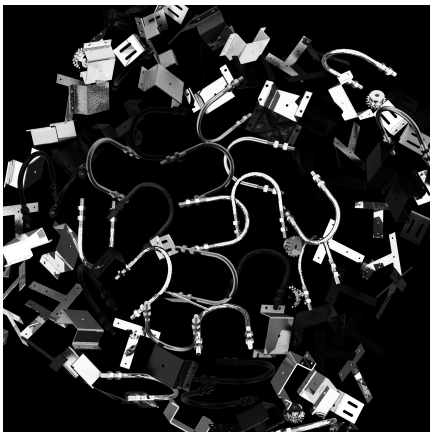
2 Dataset Analysis

Sample images from dataset

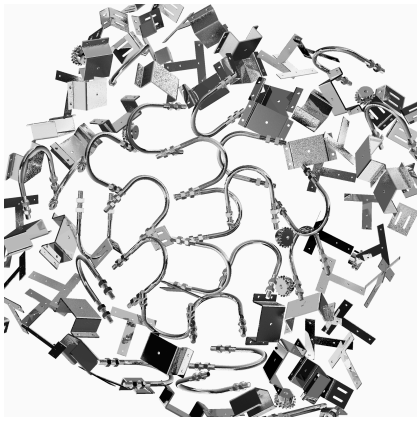
RGB Image



Object Mask Image



DOLP Image



AOLP Image

Depth Image

The original dataset includes RGB, Depth, AOLP, DOLP, and mask images. However, its structure was not optimized for training, so a custom data loader was developed to efficiently feed data into our model.

Dataset Structure:

```
DATASET_NAME
├─ camera[_CAMTYPE].json
├─ dataset_info.json
├─ test_targets_bop19.json
├─ test_targets_bop24.json
├─ [test_targets_multiview_bop25.json]
├─ models[_MODELTYPE][_eval]
│  ├─ models_info.json
│  ├─ obj_OBJ_ID.ply
├─ train|val|test[_SPLITTYPE]|onboarding_static|onboarding_dynamic
│  ├─ SCENE_ID|OBJ_ID
│  │  ├─ scene_camera[_CAMTYPE].json
│  │  ├─ scene_gt[_CAMTYPE]son
│  │  ├─ scene_gt_info[_CAMTYPE].json
│  │  ├─ scene_gt_coco[_CAMTYPE].json
│  │  ├─ depth[_CAMTYPE]
│  │  ├─ mask[_CAMTYPE]
│  │  ├─ mask_visib[_CAMTYPE]
│  │  ├─ rgb|gray[_CAMTYPE]
```

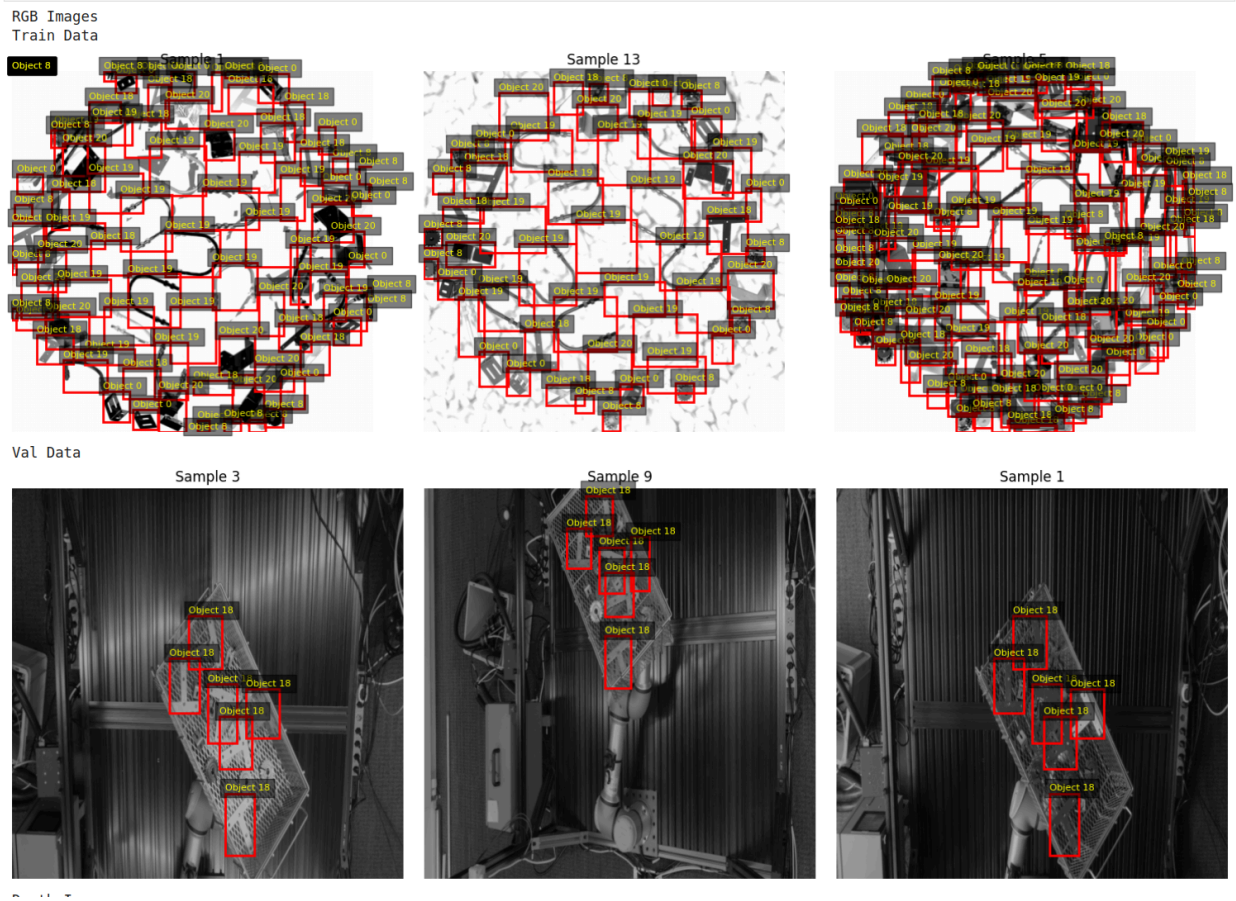
Custom Data Loader Structure:

```
DATASET_NAME
├─ Camera[_Number]
|   ├─[_Camera Type]
|   |    ├─Images
|   ├─[_Labels]
|   |    ├─BoundingBox, Object labels, 6 DoF
```

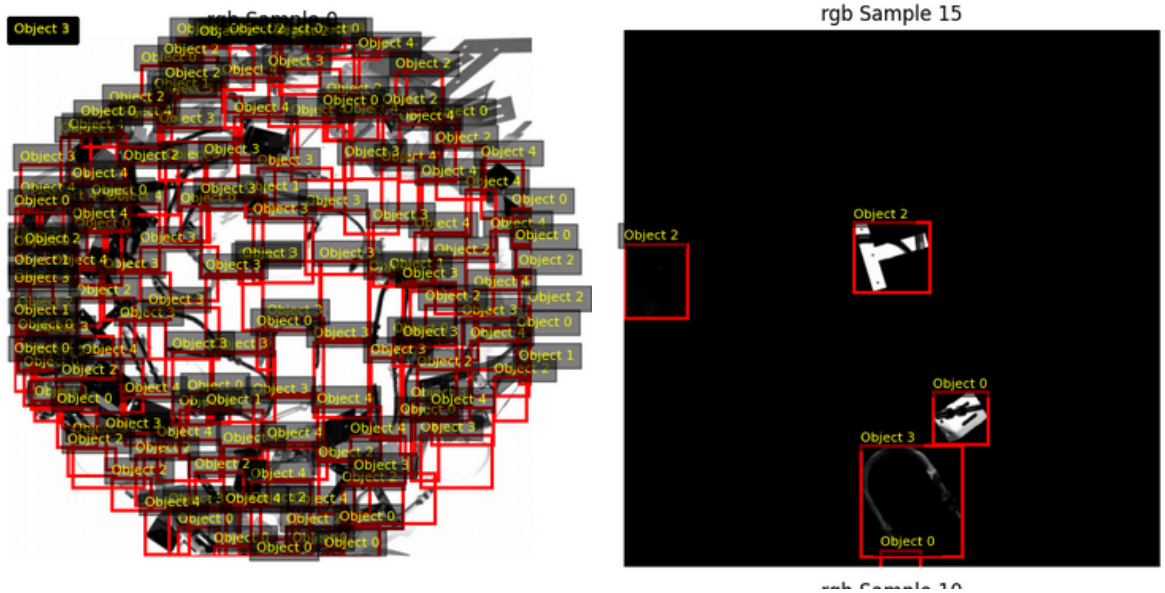
Due to the large size of the original dataset, we selected a representative subset and partitioned it into training, validation, and test splits.

Each image in the training set contains approximately 10 to 30 objects, but hardware constraints — particularly limited GPU memory — prevented us from training on full scenes with many objects simultaneously. To address this, we utilized the mask images to generate a new dataset, where each training sample is constructed by randomly selecting 5 visible object masks per image. This allowed us to train efficiently while maintaining object-level diversity within the batch.

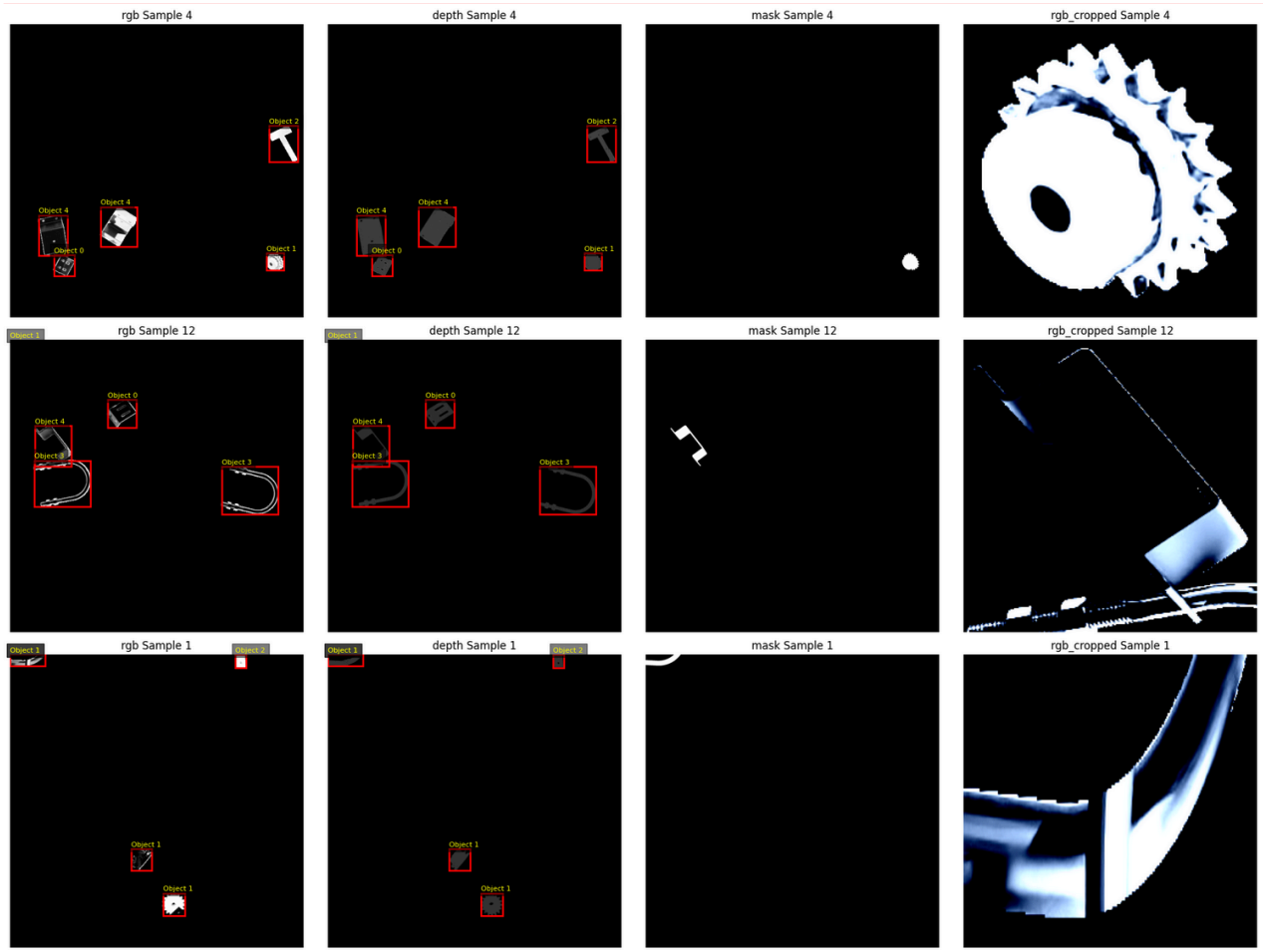
Sample images from dataset with labels



Original Image with Mask



Sample images from the new dataset



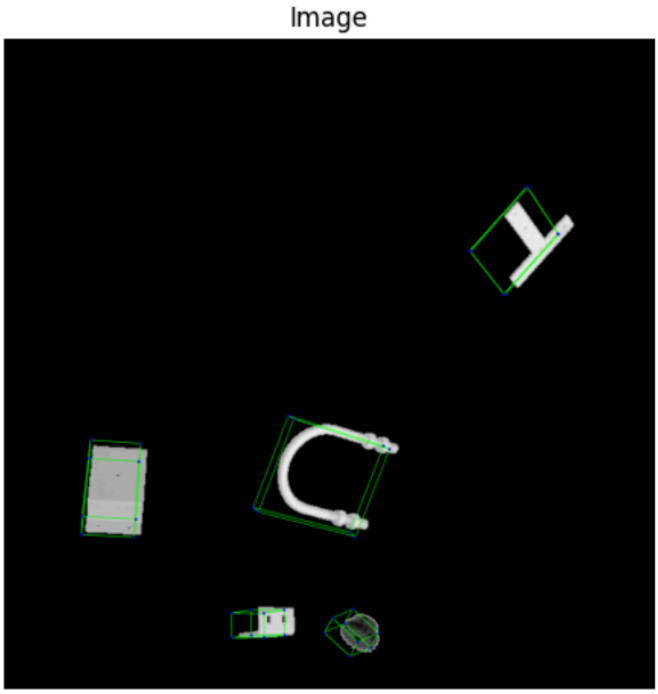
Preparing data for pose prediction:

From the selected 5 objects per image, each object is cropped and used as an input to the pose prediction model. The objective is to predict the 6 Degrees of Freedom (6-DoF) — consisting of position (X, Y, Z) and orientation (Roll, Pitch, Yaw).

The ground truth for each object is provided in the form of a rotation matrix and a translation vector. The rotation matrix is converted to Euler angles (Roll, Pitch, Yaw), and when combined with the translation vector, forms the complete 6-DoF pose representation.

Equations to get poses:
K = intrinsic camera , R = Rotation matrix , t = translation vector
X_World = 8 edge points of the object which we get using the 3d model of the object.
pose=[tx,ty,tz,roll,pitch,yaw]
projected_pts = K * [R|t] * X_world

Sample images from showcasing pose



Camera Intrinsic offset

The camera intrinsics supplied with BOP describe a 3840 × 2160 sensor, while our cropped images are 2400 × 2400, so the principal point must be shifted before any pose work. We kept the focal lengths the same, subtracted the horizontal and vertical crop offsets from the original principal point and built a new 3 × 3 matrix K, which we save alongside each frame for later reprojection.

```
# Code used for offsetting intrinsic
fx, fy, cx, cy = intr["fx"], intr["fy"], intr["cx"], intr["cy"]

orig_width = 3840
orig_height = 2160
new_width = 2400
new_height = 2400
offset = -70
# Cropping offset
crop_offset_x = ((orig_width - new_width) / 2) - offset
crop_offset_y = (orig_height - new_height) / 2

cx_new = cx - crop_offset_x
cy_new = cy  - crop_offset_y
fx_new = fx
```

```
fy_new = fy

K = np.array([[fx_new, 0, cx_new],
              [0, fy_new, cy_new],
              [0,  0,  1]], dtype=np.float32)
```

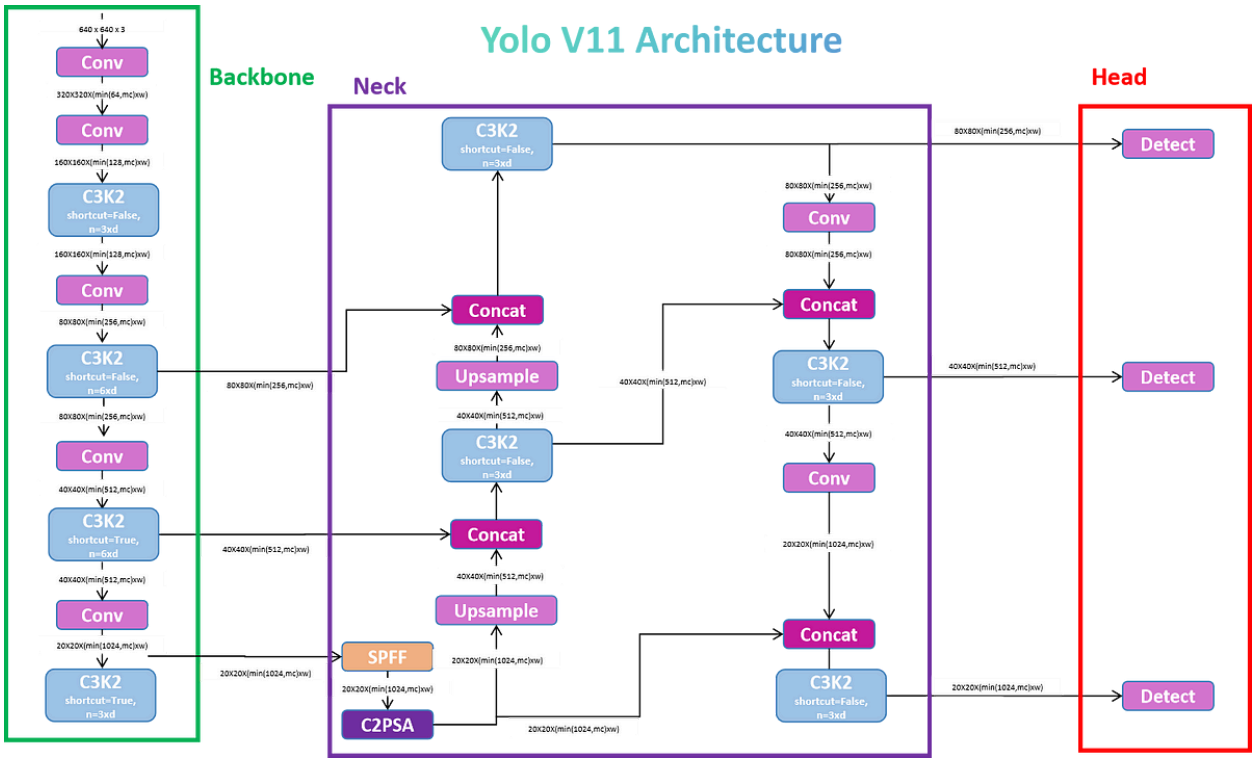
Training the YOLO MODEL

We trained our detector by extending the standard YOLO pipeline so that both the RGB image and the aligned depth map pass through the early layers of the backbone in parallel, then we concatenate the two feature tensors and feed them into the rest of YOLO's backbone, neck and detection head.

To shorten training we imported the weights from the official 10 baseline solution that already predicts one object among the different object types present in the dataset; layers whose shapes match were copied directly, giving us a strong starting point and faster convergence on our modest GPU.

The training parameters are here:

```
criterion = YOLO11Loss(reg_max=16, num_classes=10).to(device)
optimizer = optim.Adam(net.parameters(), lr=1e-4)
```



YOLO 11 LOSS

This class is a module that turns model outputs into a single training loss. It scores each object by three parts: class prediction with binary-cross-entropy (BCEWithLogitsLoss), distance distribution with distribution-focal loss (DFL), and box overlap with CIoU. These parts are weighted, summed, averaged, and returned for back-propagation.

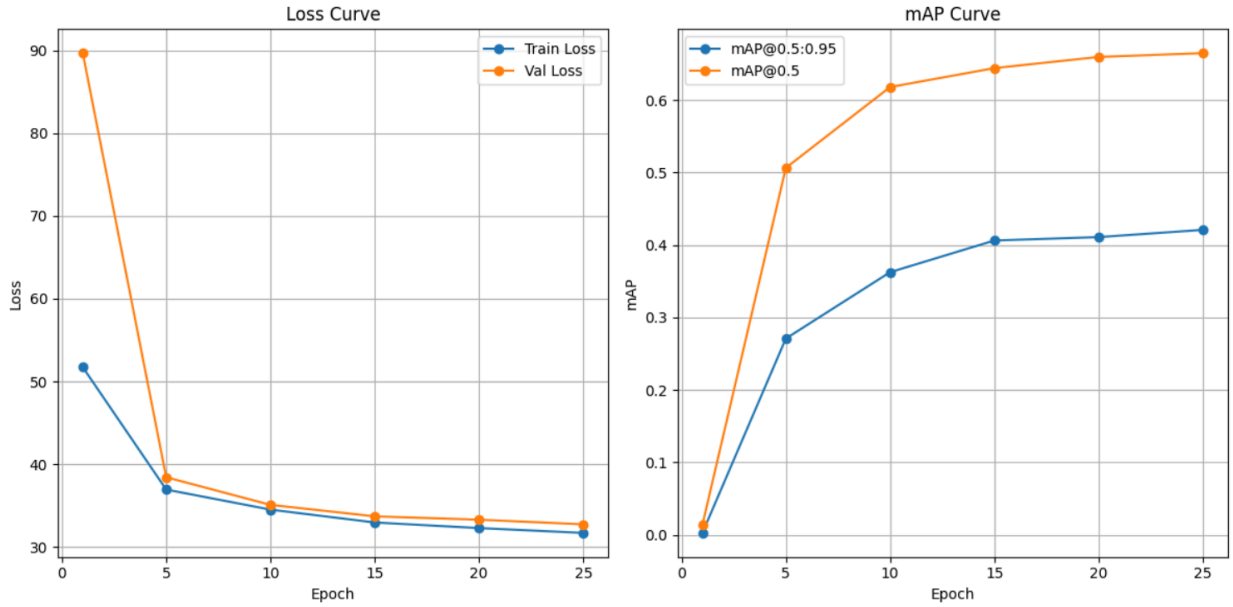
Decoder and NMS

decode_and_nms turns the network's three output maps into final detections. It converts class logits to probabilities, recovers box edges by taking the expected value of distance bins, builds boxes around each grid cell, keeps those above a confidence threshold, merges all scales, applies IoU-based non-maximum suppression, and returns the remaining boxes.

Loss too high training for all labels

Manageable loss

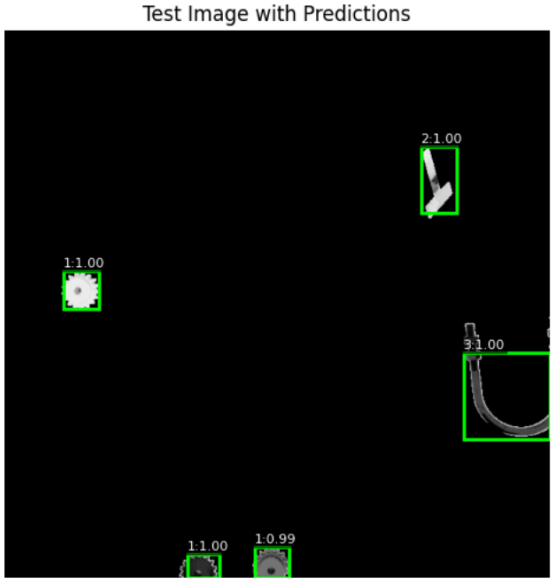
Loss and mAP plot



Training Discussion

Our training curves show rapid convergence—most gains occur within the first ten epochs, after which improvements taper off—so we concluded training at 25 epochs. Moreover, the model consistently achieves mAP scores of 0.40 and above, underscoring its strong performance.

Testing on test dataset

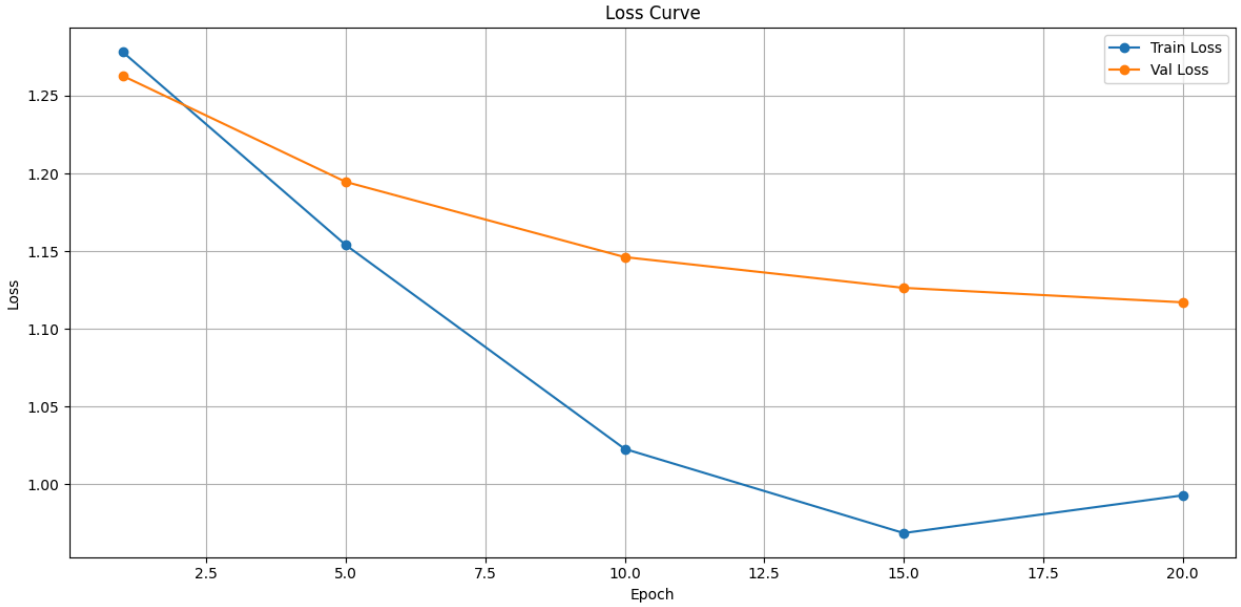


Pose Prediction Model

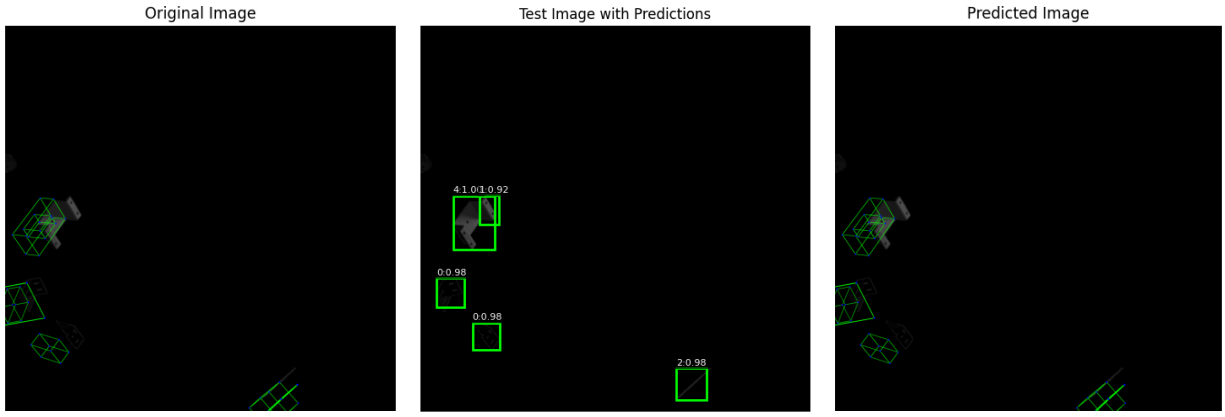
After each object is detected, we crop its RGB patch and send it through a lightweight ResNet-18 backbone. The resulting feature vector flows into two fully connected layers that output roll, pitch, yaw and three translation values. Targets come from the ground-truth rotation and translation matrices, with rotations converted to Euler angles so the network can regress six continuous numbers. We apply the updated camera intrinsic matrix before reprojection to make sure the predicted poses align with the annotations.

Finally, we have combined both models and predicted the pose of objects.

Loss curve



Model combination result



Statement of individual contribution:

Vikram Raj Nagoor Kani (VN18):

- Developed the object detection model architecture.
- Developed the custom dataset loader.
- Assisted in pose prediction model architecture.

Yudai Yamada (yyamada2):

- Assisted in developing the object detection model architecture.
- Explored research papers to find state of the art prediction models to get higher accuracy.

Sai Rohit Muralikrishnan(srm17):

- Developed the pose prediction model architecture.
- Developed the hybrid polarimetric pose estimation pipeline (including code for computing physical priors)

Harish Kumar Balaji (hkb2):

- Assisted in developing pose prediction model.

References:

1. Ultralytics, https://docs.ultralytics.com/.
2. Tao, Juneta. “RGB-D Object Detection. Depth data is widely available and… | by Juneta Tao.” Medium, 8 February 2023, https://medium.com/@juneta.tao/rgb-d-object-detection-5bbf60381c01.
3. “VainF/DeepLabV3Plus-Pytorch: Pretrained DeepLabv3 and DeepLabv3+ for Pascal VOC & Cityscapes.” GitHub, https://github.com/VainF/DeepLabV3Plus-Pytorch.
4. Gao, Daoyi, and et al. “Polarimetric Pose Prediction.” Polarimetric Pose Prediction, https://arxiv.org/pdf/2112.03810.
5. Mishra, Palash. “YOLOv11 Explained: Next-Level Object Detection with Enhanced Speed and Accuracy.”
6. https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f71.

7. Gao, D. et al. (2022). Polarimetric Pose Prediction. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds) Computer Vision – ECCV 2022. ECCV 2022. Lecture Notes in Computer Science, vol 13669. Springer, Cham. https://doi.org/10.1007/978-3-031-20077-9_43
8. Zhou X, Girdhar R, Joulin A, Krähenbühl P, Misra I. Detecting twenty‐thousand classes using image‐level supervision. *arXiv Preprint*. 2022;arXiv:2201.02605.
9. Chen J, Zhou Z, Sun M, Bao T, Zhao R, Wu L, He Z. ZeroPose: CAD‐prompted zero‐shot object 6D pose estimation in cluttered scenes. *arXiv Preprint*. 2023;arXiv:2305.17934.
10. Shafiullah NM, Paxton C, Pinto L, Chintala S, Szlam A. CLIP‐Fields: Weakly supervised semantic fields for robotic memory. *arXiv Preprint*. 2022;arXiv:2210.05663.

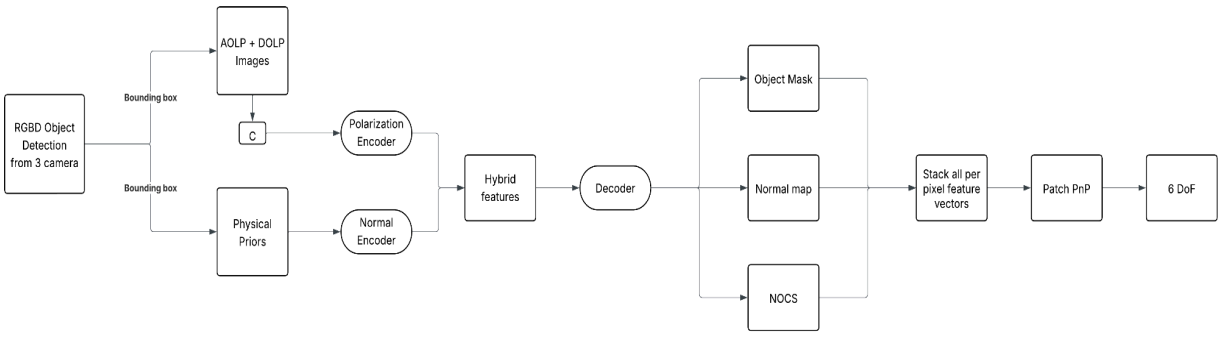
Appendix A: Alternative Approaches Tried

1.) Hybrid polarimetric pose estimation

As part of the early exploration phase, we considered implementing a physics-guided pose estimation architecture inspired by the *Polarimetric Pose Prediction* (PPP-Net) framework[4]. This approach leverages angle and degree of linear polarization (AOLP, DOLP) images to infer surface normals via Fresnel theory and feeds them into a dual-stream encoder-decoder network for 6-DoF pose estimation. Although promising, we ultimately **did not use this method in our final pipeline** due to the model's high computational and memory requirements, which exceeded our available resources.

However, significant effort went into **understanding and partially implementing the initial steps of this approach**, particularly the computation of physical priors from polarimetric data. This included writing and testing a standalone module that:

- Crops bounding boxes from AOLP and DOLP maps,
- Applies Fresnel inversion to estimate per-pixel zenith angles,
- Combines them with azimuth angles derived from AOLP to compute unit surface normals,
- Outputs diffuse and specular candidate normal maps for each object.

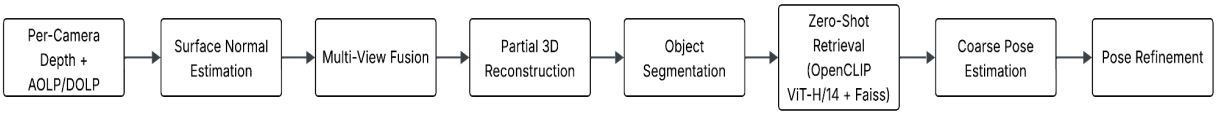


2.) Zero-Shot 6DoF Pose Estimation with CLIP and Polarization Fusion :

In the initial phase of the project, we studied recent literature on multi-modal perception and zero-shot object understanding to design a CLIP-based pipeline for 6DoF pose estimation. Inspired by works such as [8], [9], [10], we proposed a method that requires no object-specific training.

Pipeline Summary:

1. Surface Normal Estimation: Per-camera depth was fused with polarization (AOLP+DOLP) to compute refined normals, producing more reliable local geometry.
2. Multi-View Fusion: Point clouds from all cameras were aligned using extrinsics and fused into a partial 3D reconstruction using TSDF or ICP-based methods.
3. Object Segmentation: Instance masks were generated using RGB+DOLP inputs to better delineate object boundaries.
4. Zero-Shot Retrieval: Using OpenCLIP ViT-H/14, we proposed extracting embeddings from segmented object RGBs and matching them against a shape database (e.g., ShapeNet) via Faiss indexing for coarse pose retrieval.
5. Pose Refinement: Initial pose was to be refined via point-to-plane ICP or PnP, and consistency was enforced across all views



Due to lack of compute, we could not proceed with CAD model rendering, CLIP inference, or multi-view fusion at scale. No quantitative results were obtained.