3D Object Detection and Localization Using YOLO and MiDaS for Real-Time Applications

Aditya Aspat
Northeastern University
aspat.a@northeastern.edu

Jay Jajoo
Northeastern University
jajoo.j@northeastern.edu

Asmita Chetlapalli
Northeastern University
chetlapalli.a@northeastern.edu

Abstract

This study develops a vision module that employs a single monocular camera to enable real-time three-dimensional detection and localization by combining object identification with depth estimates. The system integrates the MiDaS model for accurate depth estimation and YOLOv5/YOLOv8 for efficient object detection. The proposed module ensures cost-effective and precise 3D localization, offering a viable alternative to LiDAR-based systems, which are expensive and resource-intensive. The experiments show improved detection accuracy and generalization using advanced YOLOv8 models, while MiDaS provides robust absolute depth estimation using a transformer-based DPT-Hybrid backbone.

1 Introduction

Accurate 3D object detection is essential for applications like autonomous driving. Traditional approaches relying on LiDAR are effective but expensive. This work proposes a vision-based solution combining YOLO models for object detection and MiDaS for monocular depth estimation. The method achieves high accuracy, affordability, and real-time performance.

2 Dataset and Experimental Setup

2.1 Datasets

- **KITTI Object Detection:** Includes 7,000 labeled images for training with 2D/3D annotations and nine main classes (car, van, pedestrian, cyclist, truck, misc, tram, and person sitting).
- **KITTI Depth Prediction:** Contains 500 images with paired ground truth depth maps resized to 384×384 for computational efficiency.

2.2 YOLO Object Detection Models

2.2.1 YOLOv8 Architecture

YOLOv8 improves efficiency and performance with a modular approach.

Backbone

- Employs CSPDarknet53 with enhanced Conv layers (Focus layer removed for efficiency).

Preprint. Under review.

- Introduces ELAN (Efficient Layer Aggregation Networks) to improve gradient flow.

Neck

- FPN (Feature Pyramid Network) + PANet for better multi-scale feature fusion.
- Optimized for faster computation and flexibility.

Head

- Anchor-free detection head.
- Predicts bounding boxes as $(x, y, \text{width}, \text{height})$ and class probabilities directly, reducing computation.

Both models are designed for real-time object detection, but YOLOv8 achieves better speed, accuracy, and simplicity by adopting anchor-free predictions and enhanced network modules.

2.2.2 Why YOLOv8 Over YOLOv5?

YOLOv8 offers significant improvements over YOLOv5, making it the preferred choice for object detection:

- **Anchor-Free Detection:** YOLOv8 eliminates the need for anchor boxes, simplifying training and improving performance for varying object scales.
- **Enhanced Efficiency:** Incorporates Efficient Layer Aggregation Networks (ELAN) for faster training and inference with optimized computation.
- **Improved Accuracy:** Better feature aggregation (FPN + PANet) enhances multi-scale detection, especially for small objects.
- **Modular Design:** Simplified architecture (removal of Focus layer) supports flexibility and ease of customization.
- **Better Performance:** Achieves higher precision, recall, and mAP with lower latency, ideal for real-time applications.

YOLOv8's efficiency, accuracy, and modern features make it superior for diverse object detection tasks.

2.2.3 YOLO Training Details:

YOLOv5 - Small Model (7.2M Params)

- **Training Setup:** The YOLOv5 small model was trained using the Adam optimizer instead of SGD with 350 epochs in the first experiment.
- **Modified Experiment:** Learning rate reduced to 0.001 with extended training time of 870 epochs for more robust convergence.

YOLOv8 - Medium Model (25M Params)

- **Without Data Augmentation:**
  - Optimizer: Default.
  - Epochs: 100.
  - Hyperparameters: Default settings used.
- **With Data Augmentation:**
  - Optimizer: Default.
  - Epochs: 100.
  - Data Augmentation Details:
    - HSV Augmentation: Hue = 0.014, Saturation = 0.8, Brightness = 0.4.
    - Transformations: Degrees = 2.5, Translation = 0.1, Scaling = 0.4, Shear = 1.0.

  * Flip Horizontal = 0.5, Mosaic = 0.8, Mixup = 0.1.

YOLOv8 - Large Model (43M Params)

- Model trained for 100 epochs using default hyperparameters.
- No additional data augmentation or modifications applied.

Chosen Model: For the project, we selected the YOLOv8 Medium model without data augmentation. This model provided high precision and recall, achieving a balance between accuracy and computational efficiency.

Table 1: Training configurations and results for YOLO models.

YOLO Model	Optimizer	Epochs	LR	Precision	Recall	mAP50	mAP50:90
v5-Small	Adam	350	0.01	0.6009	0.4939	0.5128	0.452
v5-Small	Adam	870	0.001	0.7824	0.4545	0.5399	0.479
v8-Med.	Auto	100	0.01	0.9418	0.8908	0.9391	0.889
v8-Med. (Data Aug.)	Auto	100	0.01	0.948	0.882	0.9350	0.885
v8-Large	Auto	100	0.01	0.9418	0.8908	0.9391	0.889

2.3 Depth Estimation Using MiDaS

2.3.1 Base Model - DPT Hybrid

Base Model: DPT_Hybrid (MiDaS Model) The depth estimation framework utilizes the **DPT_Hybrid** model, a pre-trained state-of-the-art transformer-based architecture designed for monocular depth estimation. DPT_Hybrid leverages a Vision Transformer (ViT) backbone, which efficiently captures global context and fine-grained details, making it well-suited for depth prediction tasks in diverse environments.

Key Features of DPT_Hybrid:

- **Vision Transformer Backbone:** Extracts hierarchical features by processing the image as a sequence of patches.
- **Multi-Scale Feature Aggregation:** Combines information from different spatial scales for improved depth map resolution and accuracy.
- **Pretrained Weights:** The model is initialized with pretrained weights fine-tuned on Hybrid-scale datasets like MiDaS, ensuring robust generalization.

2.3.2 Absolute Depth Model:

Initially, absolute depth estimation was performed using a linear transformation of the form:

$$\text{absolute\_depth} = \frac{\text{relative\_depth} - \text{offset}}{\text{scale\_factor}}$$

However, due to the non-linear relationship between relative and absolute depth, this approach proved insufficient for accurately capturing depth variations in complex scenes. To address this, a small neural network using 1x1 convolutional layers was introduced. These layers learn the pixel-wise non-linear transformation required to map relative depth predictions into absolute depth values, improving accuracy and adaptability without relying on manual scale and offset

To extend the capabilities of the DPT_Hybrid model for absolute depth estimation, a custom architecture wraps the MiDaS model with additional learnable components. This architecture eliminates manual scale and offset parameters and instead uses 1x1 convolutional layers to directly transform relative depth maps into absolute depth values.

Key Components of the Absolute Depth Model:

- **Relative Depth Prediction:** The DPT_Hybrid model outputs a single-channel relative depth map for input images.
- **1x1 Convolutional Transformation:** A sequence of 1x1 convolutional layers is applied to the relative depth map. These layers learn pixel-wise transformations to map relative depth values into absolute depth predictions:
  - The first convolution layer increases the feature dimension to 64 channels, enabling richer pixel-wise representations.
  - Intermediate layers apply non-linear transformations using ReLU activations.
  - The final convolution layer reduces the feature space back to 1 channel, producing the absolute depth map.
- **Resizing Step:** Before convolutional processing, the relative depth map is resized to the target resolution (e.g., 352 x 1216) using bicubic interpolation to ensure spatial alignment with ground truth maps.
- **Non-Negative Depth Enforcement:** A ReLU activation function ensures that all absolute depth values remain non-negative.

Advantages of the Architecture

- **Pixel-Wise Learning:** By using 1x1 convolutions, the model learns pixel-wise transformations efficiently without flattening the depth map, preserving spatial structure.
- **Dynamic Depth Prediction:** Instead of relying on manually tuned scale and offset parameters, the network learns the mapping from relative to absolute depth automatically.
- **Efficiency and Scalability:** The lightweight design of 1x1 convolutions ensures computational efficiency and allows the model to scale to higher resolutions and larger datasets.
- **Integration with DPT_Hybrid:** Builds on the robust relative depth predictions from the DPT_Hybrid model, combining its powerful feature extraction with an enhanced pixel-wise transformation mechanism.

2.3.3 Depth Estimation Training Process

The model is trained over **40 epochs**, focusing on iterative improvement of parameters. The training pipeline includes:

- **Loss Function:** Mean Squared Error (MSE) loss is used to compute the error between predicted depth values and ground truth depth maps, focusing only on valid (non-zero) regions. This approach is effective given the sparse nature of the ground truth depth information derived from point clouds.
- **Input and Target Preparation:** Input images and corresponding ground truth depth maps are processed in batches. Non-zero depth values are identified in the ground truth maps to ensure loss calculation focuses on valid depth data.

Loss Function The loss function is defined as:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_{\text{valid}}(i) \left(d_{\text{pred}}(i) - d_{\text{gt}}(i)\right)^2 \tag{1}$$

where d_{pred} is the predicted depth, d_{gt} is the ground truth depth, and $\mathbf{1}_{\text{valid}}$ ensures that the loss is calculated only for valid, non-zero depth regions.

2.4 3D Position Estimation Using Camera Intrinsics, YOLOv8, and Depth Model

To estimate the 3D position of detected objects relative to the camera frame, we use the camera intrinsic matrix K, the 2D pixel coordinates (x, y) from YOLOv8 bounding boxes, and the depth values Z predicted by the depth estimation model.

Camera Intrinsics Matrix The camera intrinsic matrix K is given by:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}.$$

Here:

- f_x, f_y: Focal lengths along the x and y axes (in pixels).
- c_x, c_y: Principal point coordinates (optical center of the image).

2D Pixel Coordinates from YOLOv8 The YOLOv8 model detects objects in the image and provides bounding boxes. For each bounding box, the center pixel coordinates (x, y) of the object are extracted:

$$(x, y) = (\text{center\_x}, \text{center\_y}).$$

Depth Value from Depth Model The depth model predicts a depth value Z for each pixel, representing the distance between the camera and the object along the z-axis.

3D Position Reconstruction Using the camera intrinsic matrix K, the 2D pixel coordinates (x, y), and the depth value Z, the 3D position (X, Y, Z) of the object in the camera frame is computed as:

$$X = \frac{(x - c_x) \cdot Z}{f_x}, \quad Y = \frac{(y - c_y) \cdot Z}{f_y}, \quad Z = Z.$$

Here:

- x, y: 2D pixel coordinates of the object center (from YOLOv8).
- Z: Depth value at the pixel location (from the depth model).
- c_x, c_y: Principal point coordinates.
- f_x, f_y: Focal lengths of the camera.

Combining Class Labels and 3D Positions For each detected object, the following information is generated:

- Class label (e.g., "car", "person") from YOLOv8.
- 3D position (X, Y, Z) relative to the camera frame.

Workflow Summary

1. Extract 2D pixel coordinates (x, y) and class labels from YOLOv8 detections.
2. Obtain the depth value Z for each pixel from the depth estimation model.
3. Use the camera intrinsic matrix K to compute the 3D position:

$$X = \frac{(x - c_x) \cdot Z}{f_x}, \quad Y = \frac{(y - c_y) \cdot Z}{f_y}, \quad Z = Z.$$

4. Combine the class labels and 3D positions into a structured output.

Result The final output consists of the class label and 3D position (X, Y, Z) for each detected object, providing their location in the camera coordinate frame.

3 Results

3.1 Key Findings on Object Detection task

- **YOLOv8 Superiority:** Anchor-free head and improved architecture led to better detection performance compared to YOLOv5.

- **Medium vs Hybrid Models:** The KITTI dataset complexity limited improvements for Hybridr models due to diminishing returns.
- **Effect of Data Augmentation:** Minimal improvements as YOLOv8 already includes robust built-in augmentation methods.
- **Class Imbalance:** Classes like "person sitting" and "pedestrian" showed lower mAP due to fewer samples and occlusion challenges.

3.2 Key Findings on Depth Estimation Task

- **Transformation:** Initially, a simple linear transformation was applied to compute absolute depth from the relative depth values predicted by the MiDaS model. However, due to the inherent non-linear relationship between relative and absolute depth, the performance was limited. This non-linearity became evident during training, where the loss plateaued after a certain point.
- **Model Update:** To address this, a small neural network using 1x1 convolutional layers was introduced to learn the pixel-wise non-linear transformation from relative to absolute depth. This approach allowed the model to adapt dynamically to the depth variations in the scene.
- **Loss:** The overall loss for an image was approximately 130 meters, indicating that, on average, the absolute depth values deviate by 130 meters compared to the ground truth, per image. This highlights the need for further optimization or additional enhancements in the depth prediction pipeline.

3.3 Overall Feedback on This Approach

Depth information reducing overall Accuracy: YOLO provides highly accurate pixel coordinates, which can effectively predict the x and y positions. However, inaccuracies in the depth measurements for certain points in the image result in erroneous 3D world space coordinates. The integration of YOLO's output with depth information requires more robust depth estimation to improve 3D localization.

3.4 Visualization

3.4.1 YOLO Model Output

Figure 1: Original image

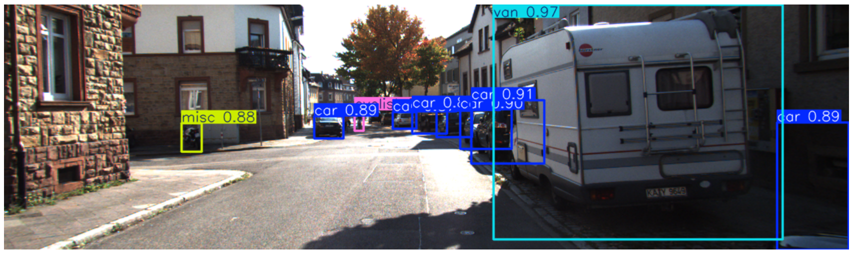

Figure 2: Detected objects with class labels, probabilies, and bounding boxes.

3.4.2 Depth Estimation Output

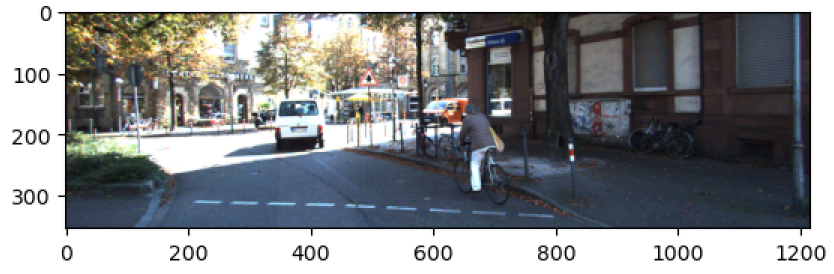
Figure 3: Original image

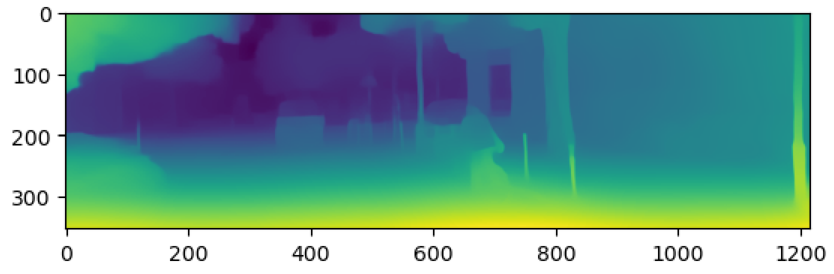
Figure 4: Inverse relative depth from MiDaS

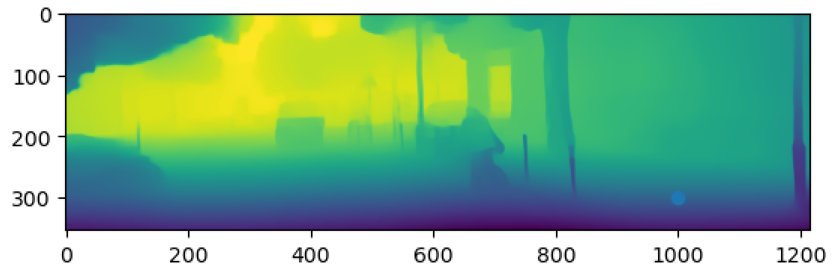
Figure 5: Absolute depth from our model

3.4.3 Final Output

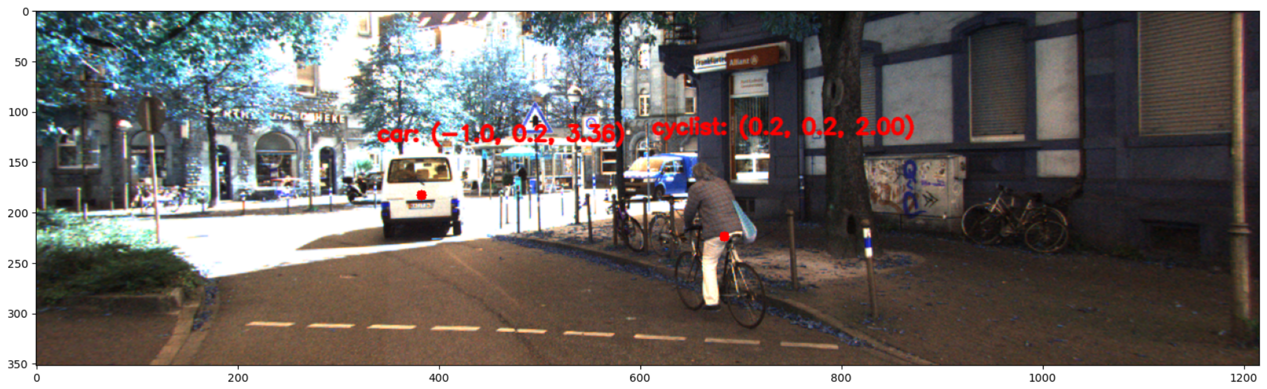

Figure 6: Localization of objects in 3D space

4 Conclusion

This study integrates YOLO and MiDaS to deliver a real-time, cost-effective monocular solution for 3D perception. While the results highlight YOLOv8's architectural strengths in object detection and MiDaS's robust depth estimation capabilities, certain limitations remain. The depth predictions exhibit inaccuracies due to the non-linear relationship between relative and absolute depth, which affects the precision of 3D localization. Additionally, errors in depth estimation can propagate to the final 3D coordinates, limiting performance in complex or dynamic scenes. Despite these flaws, the method shows promise for applications like autonomous driving, particularly where cost and real-time operation are critical considerations.

References

1. KITTI Dataset: `http://www.cvlibs.net/datasets/kitti/`
2. MiDaS: `https://github.com/isl-org/MiDaS`
3. Ultralytics YOLOv8 Documentation.
4. YOLO (You Only Look Once) Research Paper: `https://arxiv.org/abs/1506.02640`