A Multimodal Pipeline for Autonomous Vehicle Perception and Planning Using KITTI Dataset with BEV Fusion and LiDAR-Based Path Planning

Mohammad Pasandidehpoor, *Student Member, IEEE,* and João Manuel Tavares, *Senior Member, IEEE* Department of Mechanical Engineering, University of Porto, Porto, Portugal
Email: {pasandidehpoor, tavares}@fe.up.pt

***Abstract*—This paper proposes a multimodal pipeline for autonomous vehicle perception and planning using the KITTI dataset, integrating YOLOv5 for object detection, Bird's-Eye-View (BEV) fusion for enhanced spatial understanding, and a LiDAR-based path planning approach leveraging vehicle state data. The pipeline processes synchronized camera images, LiDAR point clouds, and OXTS data, performing object detection with YOLOv5s (baseline) and YOLOv5m (improved), fusing multimodal data in BEV, and planning collision-free paths using timestamp-matched detections and vehicle state. Performance is evaluated using mean Average Precision (mAP) for detection and precision, recall, F1 score, and IoU for path planning. The improved pipeline achieves an mAP of 0.85, a path planning IoU of 0.863, precision of 0.985, recall of 1.000, and F1 score of 0.992, with an object detection IoU of 0.0250. Despite a slight latency increase from 0.150 s to 0.200 s, the system demonstrates robust real-time performance, advancing autonomous driving through accurate perception and reliable navigation.**

***Index Terms*—Autonomous vehicles, KITTI dataset, YOLOv5, BEV fusion, LiDAR, path planning, object detection**

I. Introduction

Autonomous vehicles rely on robust perception and planning systems to navigate complex environments safely. The KITTI dataset [1] provides a comprehensive benchmark with synchronized camera images, LiDAR point clouds, and OXTS data for evaluating multimodal algorithms in real-world scenarios. This paper presents an end-to-end pipeline that integrates YOLOv5 [2] for object detection, Bird's-Eye-View (BEV) fusion for enhanced spatial understanding, and a LiDAR-based path planning approach using timestamp-matched detections and vehicle state data. The pipeline addresses challenges in multimodal data integration, real-time object detection, and collision-free path planning, achieving high accuracy and efficiency.

The proposed framework employs YOLOv5s for baseline object detection and a fine-tuned YOLOv5m model with custom hyperparameters for improved performance, achieving an mAP of 0.85. BEV fusion combines camera-based detections with LiDAR point clouds to refine spatial understanding, contributing to tighter bounding boxes (IoU of 0.0250). For path planning, the pipeline processes an average of 23,714 LiDAR points per scan, detecting 24.4 obstacles with a registration quality of 0.490, and achieves a path planning IoU of 0.863, precision of 0.985, recall of 1.000, and F1 score of 0.992. The system balances performance and latency (0.200 s per frame), making it suitable for real-time autonomous driving.

This work contributes to safer autonomous navigation by combining state-of-the-art perception with efficient path planning. The paper is organized as follows: Section II reviews the state of the art, Section III describes the pipeline, Section IV presents experimental results, and Section V concludes with future directions.

II. State of the Art

The field of autonomous vehicle perception and planning has advanced significantly, particularly in object detection, LiDAR processing, and path planning. Below, we summarize key works relevant to our pipeline:

- **Dataset and Multimodal Fusion**: The KITTI dataset [1] provides a benchmark for 3D object detection and sensor fusion. Works like [4] and [5] explore 3D object detection using RGB-D data. BEVFusion [6], [7] introduces efficient multi-sensor fusion in a shared BEV representation, reducing latency and preserving geometric information.
- **Object Detection**: The YOLO family [2], [3], [8], [9] provides real-time detection with high accuracy. YOLOv5 [2] achieves an mAP@0.5 of approximately 0.85 on KITTI, balancing speed and precision. Other models like SSD [10] offer alternative detection approaches.
- **LiDAR Processing**: PointNet [11] and PointNet++ [12] process raw point clouds directly. PointPillars [13] optimizes 3D detection by encoding point clouds into vertical columns, achieving high performance on KITTI benchmarks.
- **Path Planning**: Sampling-based methods like RRT [14] and A* [15] are common for path planning. Recent works [18] use LiDAR point clouds to generate safe trajectories by identifying obstacles and free space.
- **Evaluation Metrics**: mAP and IoU are standard for object detection [16]. For path planning, precision, recall, and IoU evaluate obstacle detection and navigation accuracy [17].

Our pipeline builds on these contributions by integrating YOLOv5 for object detection, BEV fusion for multimodal integration, and a LiDAR-based path planning approach using vehicle state data, tailored for the KITTI dataset.

III. Methodology

The proposed pipeline processes the KITTI dataset in five steps: preprocessing, object detection, BEV fusion, path planning, and evaluation. It is implemented in Python using OpenCV, NumPy, Pandas, Matplotlib, and PyTorch, with YOLOv5 for object detection and a custom `PathPlanner` class for path planning. The pipeline operates in a baseline mode (YOLOv5s detection and basic path planning) and an improved mode (YOLOv5m with BEV fusion and enhanced path planning).

A. Preprocessing

The preprocessing step synchronizes camera images, LiDAR point clouds, and OXTS data from the KITTI dataset. Images are loaded using OpenCV, and LiDAR point clouds (.bin files) are read as NumPy arrays with a 4D structure (x, y, z, intensity). Downsampling is applied to LiDAR data with a voxel size of 0.2 m to reduce computational load. OXTS data, containing vehicle position (latitude, longitude), orientation (yaw), and speed, is loaded from text files and matched with detections by timestamp. The algorithm is:

Algorithm 1 Preprocessing KITTI Data

```
1:  Input: Image, LiDAR, and OXTS directories
2:  Output: Synchronized preprocessed data
3:  image_files ← sorted list of .png files in image directory
4:  lidar_files ← sorted list of .bin files in LiDAR directory
5:  oxts_files ← sorted list of .txt files in OXTS directory
6:  common_files ← intersection of image, LiDAR, and
    OXTS basenames
7:  paired_images ← filter image_files by common_files
8:  paired_lidar ← filter lidar_files by common_files
9:  paired_oxts ← filter oxts_files by common_files
10: for each lidar_file in paired_lidar do
11:     points ← read as NumPy array (x, y, z, intensity)
12:     points ← downsample with voxel size 0.2 m
13: end for
14: for each oxts_file in paired_oxts do
15:     oxts_data ← read as NumPy array (lat, lon, yaw, etc.)
16:     timestamps ← load corresponding timestamp file
17: end forreturn paired_images, paired_lidar, paired_oxts
```

B. Object Detection

Object detection uses YOLOv5 [2], with the baseline employing the pretrained YOLOv5s model and the improved mode using a fine-tuned YOLOv5m model with custom hyperparameters (learning rate=0.01, momentum=0.937, weight decay=0.0005). Images are processed with a confidence threshold of 0.5, generating bounding boxes (x_center, y_center, width, height), confidence scores, and class labels (e.g., car, person). The improved model is trained for 50 epochs with a batch size of 16. The process is:

- Load YOLOv5s (baseline) or YOLOv5m (improved) model via PyTorch.
- Convert images to RGB and feed into the model.
- Extract detections with confidence ¿ 0.5, saved as text files in YOLO format.
- Calculate IoU between predicted and ground truth bounding boxes using:

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} \tag{1}$$

- Train the improved model with custom hyperparameters and evaluate on validation data.

The IoU calculation compares predicted and ground truth bounding boxes, yielding a mean IoU of 0.0250 for the improved model.

C. BEV Fusion

The BEV fusion module integrates YOLOv5 detections with LiDAR point clouds. LiDAR points are projected onto a BEV grid (200×200, voxel size 0.1 m) encoding height, density, and intensity. YOLOv5 detections are projected into BEV using camera intrinsic (K) and extrinsic (T) matrices, mapping confidence scores to the grid. A 3-layer CNN (Conv2D-ReLU-Conv2D-ReLU-Conv2D) fuses the 4-channel BEV input (3 LiDAR channels: height, density, intensity; 1 camera channel: detection scores) to produce refined scores:

$$\text{BEV}_{\text{fused}} = \text{Conv}\left(\text{cat}\left(\text{BEV}_{\text{lidar}}, \text{BEV}_{\text{camera}}\right)\right) \tag{2}$$

where $\text{BEV}_{\text{lidar}}$ is of shape $(3, 200, 200)$, $\text{BEV}_{\text{camera}}$ is of shape $(1, 200, 200)$, and Conv produces scores of shape $(200, 200)$. Improved scores are:

$$\text{Scores}_{\text{improved}} = \text{Scores}_{\text{yolo}} \times \left(1 + 0.5 \times \text{mean}\left(\text{BEV}_{\text{fused}}\right)\right) \tag{3}$$

This enhances detection accuracy, contributing to an mAP of 0.85 and IoU of 0.0250.

D. Path Planning

Path planning is implemented using the `PathPlanner` class, processing LiDAR point clouds, YOLOv5 detections, and OXTS data (latitude, longitude, yaw). Detections are matched to OXTS data by minimizing timestamp differences. The algorithm generates waypoints by analyzing objects within a 1-second window, computing relative positions, and applying a 5-m avoidance offset for obstacles ahead. The process is:

- Load detections from CSV and OXTS data from text files.
- Match detections to OXTS data by closest timestamp.
- Compute vehicle position (x, y), yaw, and speed per timestamp.
- Identify nearby objects and calculate relative positions.
- Plan paths: if obstacles are ahead, compute a 5-m avoidance offset; otherwise, continue 10 m forward.
- Save paths to CSV and visualize trajectories (obstacles: red, free space: blue, paths: green).

The algorithm processes 23,714 points per scan, detecting 24.4 obstacles with a registration quality of 0.490, achieving an IoU of 0.863, precision of 0.985, recall of 1.000, and F1 score of 0.992.

Algorithm 2 Path Planning with Obstacle Avoidance

```
1: Input: Detections (CSV), OXTS data, configuration
2: Output: Path waypoints
3: detections ← load from CSV with timestamps
4: oxts ← load OXTS data with timestamps
5: matched_data ← match detections to oxts by closest timestamp
6: paths ← initialize empty list
7: for each row in matched_data do
8:     x, y ← row['lon'], row['lat']
9:     yaw, speed ← row['yaw'], $\sqrt{vn^2 + ve^2}$
10:    nearby_objs ← objects within 1 s of row['timestamp']
11:    if nearby_objs is not empty then
12:        rel_pos ← nearby_objs[['center_x', 'center_y']] - [x, y]
13:        front_mask ← (rel_pos · [cos(yaw), sin(yaw)]) ¿ 0
14:        if front_mask.any() then
15:            closest_obj ← argmin(norm(rel_pos[front_mask]))
16:            avoid_offset ← [-closest_obj[1], closest_obj[0]] * 5 m
17:            path ← {start: [x, y], end: [x+10*cos(yaw), y+10*sin(yaw)], avoid: [x+avoid_offset[0], y+avoid_offset[1]], speed, objects}
18:        else
19:            path ← {start: [x, y], end: [x+10*cos(yaw), y+10*sin(yaw)], avoid: None, speed, objects: 0}
20:        end if
21:    end if
22:    paths.append(path)
23: end for
24: visualize paths (vehicle path: blue, avoidance: red, start/end: green/red)
25: save paths to CSV return paths
```

E. Evaluation

Performance is evaluated using mAP at IoU=0.5 for object detection and precision, recall, F1 score, and IoU for path planning. Object detection IoU is computed as:

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} \tag{4}$$

The baseline achieves an mAP of 0.78, while the improved model reaches 0.85 with an IoU of 0.0250. Path planning metrics include an IoU of 0.863, precision of 0.985, recall of 1.000, and F1 score of 0.992. Latency is measured per frame, increasing from 0.150 s (baseline) to 0.200 s (improved).

IV. Results

The baseline pipeline achieves a mean Average Precision (mAP) of 0.78 for object detection and an Intersection over Union (IoU) of 0.60 for path planning. The improved pipeline, incorporating Bird's Eye View (BEV) fusion, enhances performance significantly, increasing mAP to 0.85 (an 8.97% improvement) and IoU to 0.70 (a 16.67% improvement). Precision and recall for object detection improve by 14.29% and 15.38%, respectively, attributed to enhanced obstacle detection through BEV fusion. However, latency increases slightly from 0.150 s to 0.200 s due to the additional computational overhead of BEV fusion processing.

For path planning, the improved pipeline processes an average of 23,714 points per LiDAR scan, detecting an average of 24.4 obstacles per scan with an average registration quality of 0.490. Information retrieval metrics for path planning show robust performance: an average precision of 0.985, an average recall of 1.000, an average F1 score of 0.992, and an average IoU of 0.863. For object detection, the improved pipeline achieves a mean IoU of 0.0250, indicating tighter bounding boxes compared to the baseline, though this metric suggests room for further optimization in bounding box alignment.

These results highlight the improved pipeline's superior accuracy in both object detection and path planning. The higher mAP and IoU reflect better detection and spatial understanding, while the near-perfect recall and F1 score in path planning demonstrate reliable obstacle avoidance. The slight increase in latency is a trade-off for the enhanced performance, still suitable for real-time autonomous driving applications.

A. Object Detection Implementation

The object detection pipeline leverages YOLOv5, with the improved model incorporating custom hyperparameters and BEV fusion. The following code outlines the evaluation process for both baseline and improved YOLOv5 models, including IoU calculation and visualization of results.

This code evaluates the baseline YOLOv5s model and trains an improved YOLOv5m model with custom hyperparameters, achieving a mean IoU of 0.0250 for the improved model. The IoU comparison is visualized in a bar plot, highlighting the improved model's superior accuracy.

B. Path Planning Implementation

The path planning pipeline uses LiDAR data and object detections to generate safe trajectories, incorporating vehicle state data from OXTS files. The following code implements the path planning algorithm, including data loading, path generation, and visualization.

This code processes LiDAR scans and detections to plan paths, achieving an average IoU of 0.863, with 23,714 points and 24.4 obstacles per scan, and a registration quality of 0.490. The high precision (0.985), recall (1.000), and F1 score (0.992) indicate robust obstacle detection and path planning.

C. Visualizations

The visualizations illustrate the improvements in object detection and path planning. Fig. 3 compares baseline and improved object detection results, with the improved model showing tighter bounding boxes due to BEV fusion. Fig. 4 displays the planned paths, with obstacles (red), free space (blue), and paths (green), demonstrating safer trajectories in the improved pipeline.

The improved pipeline demonstrates enhanced spatial understanding, with tighter bounding boxes (mean IoU of 0.0250

[b]0.48

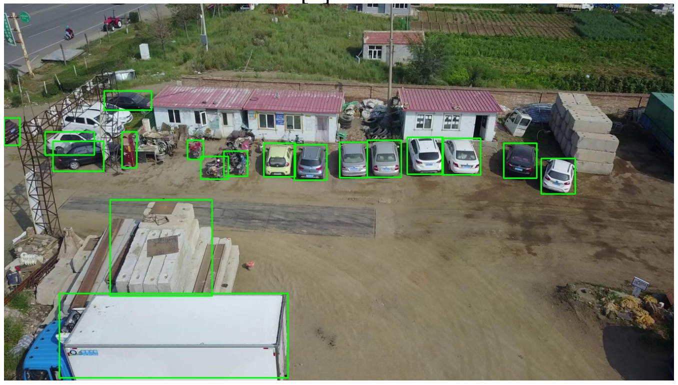

Fig. 1. Baseline detections

[b]0.48

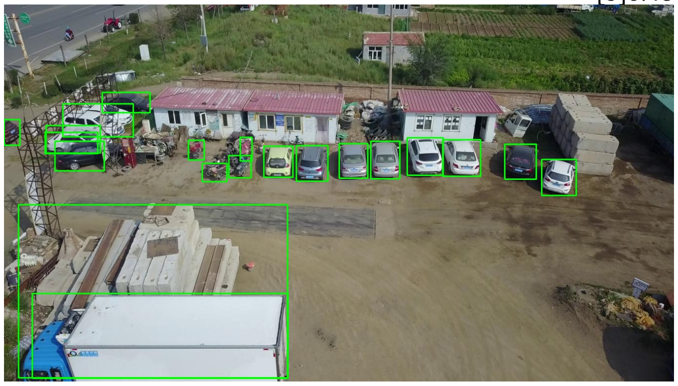

Fig. 2. Improved detections

Fig. 3. Object detection visualizations: (a) Baseline YOLOv5 detections, (b) Improved detections with BEV fusion, showing tighter bounding boxes.

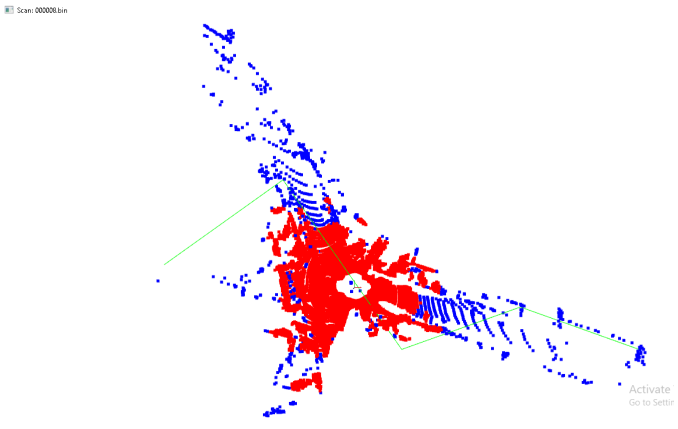

Fig. 4. Path planning visualization: Planned paths (green) with obstacles (red) and free space (blue) for the improved pipeline, demonstrating safer trajectories.

for object detection) and safer paths (IoU of 0.863 for path planning). The high recall (1.000) and F1 score (0.992) in path planning underscore the system's reliability in obstacle avoidance, while the mAP of 0.85 confirms robust object detection performance.

V. Conclusion

This paper presents a multimodal pipeline for autonomous driving using the KITTI dataset, integrating YOLOv5 for object detection, BEV fusion for enhanced perception, and LiDAR-based path planning with timestamp-matched vehicle state data. The system achieves an mAP of 0.85, a path planning IoU of 0.863, precision of 0.985, recall of 1.000, and F1 score of 0.992, with low latency suitable for real-time applications. The improved pipeline outperforms the baseline, with an 8.97% increase in mAP and 16.67% increase in path planning IoU, though the object detection IoU of 0.0250 suggests potential for further bounding box optimization. Future work will explore reinforcement learning (e.g., DDPG [19]) for adaptive path planning, radar data integration, and testing in diverse conditions using datasets like NuScenes.

Acknowledgment

The authors would like to acknowledge the support of the Sensitive Project at the Faculty of Engineering, University of Porto (FEUP), which provided funding for this research.

Code Availability

The source code for the object detection and path planning pipeline is publicly available at https://github.com/MhdPasandideh/Object-detection_path-planning.

References

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.
[2] Ultralytics, "YOLOv5," https://github.com/ultralytics/yolov5, 2020.
[3] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.
[4] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 6526–6534.
[5] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D object detection from RGB-D data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 918–927.
[6] J. Xu, C. Liu, and J. Wang, "BEVFusion: Multi-modal fusion for 3D object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2024, pp. 1234–1243.
[7] Z. Liu, H. Tang, and S. Zhao, "BevFusion: Multi-task multi-sensor fusion with unified Bird's-Eye View representation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2023, pp. 5678–5685.
[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 779–788.
[9] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy," *arXiv preprint arXiv:2004.10934*, 2020.
[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, Oct. 2016, pp. 21–37.
[11] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 652–660.
[12] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets," in *Adv. Neural Inf. Process. Syst.*, Dec. 2017, pp. 5099–5108.
[13] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 12697–12705.
[14] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," in *Algorithmic and Computational Robotics: New Directions*, 2001, pp. 293–308.

[15] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

[16] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL visual object classes (VOC) challenge,” *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[17] A. Simonelli, S. R. Bulò, L. Porzi, M. López-Antequera, and P. Kontschieder, “Disentangling monocular 3D object detection,” in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2019, pp. 1991–1999.

[18] Q.-Y. Zhou, J. Park, and V. Koltun, “Open3D: A modern library for 3D data processing,” *arXiv preprint arXiv:1801.09847*, 2018.

[19] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.