Monocular 3D Object Detection

Aaditya Gala*, Connor Blandin†, Rabia Konuk‡
*†‡University of Michigan, Ann Arbor, USA
Email: aadityag@umich.edu, cblandin@umich.edu, rkonuk@umich.edu

***Abstract*—In the rapidly evolving field of autonomous vehicle technology, monocular 3D object detection presents a challenging yet crucial frontier, particularly for understanding driving environments through limited sensory data. This provides the opportunity for more redundancy or more economical solutions to bring products to the market. This study explores this domain using the innovative MonoCon (Monocular Contexts) framework, which negates the need for supplemental inputs like depth sensors. Our approach, centered on MonoCon's unique anchor-offset formulation, represents a significant step forward in deriving 3D spatial information from 2D images. Employing the KITTI 3D Object Detection Dataset, we rigorously addressed dataset conversion and calibration challenges, ensuring robust model training and validation. Our model's achievement of a 9.6555 Average Precision (AP) score at 300 epochs marks a substantial advancement over existing benchmarks in this domain, underscoring MonoCon's potential to enhance monocular 3D object detection in autonomous vehicles. Future work may focus on extending these findings to more diverse driving environments and exploring the integration of MonoCon with other autonomous system components.**

***Index Terms*—Autonomous Vehicles, Monocular 3D Object Detection, Neural Networks, Semantic Segmentation, Temporal Data Handling**

I. Introduction

Advancements in autonomous vehicle technology hinge crucially on the capability to perceive and understand the driving environment with precision. This project focuses on one of the pivotal challenges in this domain: monocular 3D object detection. This technique, which relies on information from a single camera, is a cost-effective and practical solution for environments where deploying multiple sensors is unfeasible [1]. However, it brings unique challenges, particularly in accurately deciphering depth and spatial relationships from two-dimensional image data [2].

Monocular 3D object detection is crucial for a range of computer vision applications, notably autonomous driving and robot navigation. While high-performing methods in this field often necessitate expensive setups like LIDAR sensors, monocular 3D object detection presents an appealing alternative due to its potential for reduced costs and increased modular redundancy [3]. This project is geared towards 3D object detection within the scope of autonomous driving, aiming to estimate the 3D bounding box for object instances, such as cars, from 2D images. This estimation process is fundamental to the KITTI benchmark, a standard in autonomous vehicle research [4].

To address the complexities of monocular 3D detection without relying on extra information like depth sensors or multiple frames, we have chosen the MonoCon (Monocular Contexts) framework as our foundation. MonoCon stands out with its anchor-offset formulation, which enhances the prediction of 3D centers based on 2D bounding box anchors. This formulation underpins a method that significantly bolsters overall performance, making MonoCon an ideal foundation for our project's objectives.

In practice, MonoCon employs a streamlined design, encompassing a Deep Neural Network (DNN) based feature backbone and several regression head branches. These branches are integral to learning the essential parameters for 3D bounding box prediction and the auxiliary contexts necessary for monocular detection [5]. This approach ensures that our project not only navigates the inherent challenges of monocular detection but also innovates within them, aiming to enhance the applicability of the MonoCon framework to our specific dataset focused on the 'Car' class. Such innovation is not merely a technical adaptation but a stride towards contributing to the broader field of autonomous driving, demonstrating the potential of monocular 3D object detection in shaping the future of autonomous navigation systems.

II. Methodology

A. Data Handling

Fig. 1. Example Training Image

1) Dataset Acquisition: The cornerstone of our project is the employment of the KITTI Dataset Format, a widely used and easily transferable format. The dataset is based on a multi-agent realistic simulation, that offers the ability to construct complicated dynamic situations, with many targets varying levels of occlusion behind other targets and objects in the scene. Figure 1 is an example training image. The dataset was produced in a different format and had to be converted into the KITTI format. Key components of this dataset include:

- Calibration Files (calib): These files are vital for precise camera calibration. Accurate calibration is indispensable in 3D object detection and pose estimation, as it ensures

that the spatial relationships captured in 2D images are correctly interpreted in three dimensions.
- RGB Images (image_2): These images form the primary input for our detection model. They provide the visual data necessary for the model to identify and interpret the various elements within the driving environment.
- Label Files (label_2): These files furnish annotations for the training images. They include both 2D and 3D bounding box data along with object classes, which are crucial for training the model to accurately detect and categorize objects within its field of vision.

2) Dataset Conversion: To ensure compatibility with the MonoCon model, the data set was converted into the popular Kitti format, which is what the MonoCon architecture required. This conversion process was streamlined by utilizing helper scripts from the MonoCon repository, which performed the following functions:

- Format Alignment: The scripts facilitated the proper alignment of the dataset format, ensuring it was suitable for both the training and evaluation phases of the model. This step is crucial for maintaining the integrity of the data throughout the model's learning process.
- Calibration Data Maintenance: Alongside format alignment, these scripts also ensured the accurate maintenance of calibration data in relation to the corresponding images and annotations. This aspect is critical for ensuring that the spatial information inherent in the 3D object detection process is accurately captured and utilized.

The meticulous approach to data handling, characterized by these two key processes, laid a robust foundation for subsequent phases of model training and evaluation. It exemplified a systematic and thorough methodology, reflecting the scientific rigor necessary for advancing in the field of autonomous vehicle technology.

B. Model Description

1) Architecture Overview: The MonoCon architecture is elegantly simple but effective, comprising three main components:

- **Feature Backbone**: This backbone processes input RGB images (dimensions $3 \times H \times W$) to compute an output feature map of dimensions $D \times h \times w$, where D is the output feature map dimension and h and w are determined by the overall stride/sub-sampling rate s in the backbone (e.g., $s = 4$). This design enables efficient feature learning and is integral to the depth and spatial processing required for 3D object detection.
- **3D Bounding Box Regression Heads**: MonoCon employs an anchor-offset formulation for learning the projected 3D bounding box center in the image plane. This regression head computes a class-specific heatmap for the 2D bounding box center. It then defines the offset vector from the 2D bounding box center to the projected 3D bounding box center, crucial for accurate 3D object localization.
- **Auxiliary Context Regression Heads**: A distinctive feature of MonoCon is its use of auxiliary contexts as learning tasks, exploiting projection information from 3D bounding boxes. This includes heatmaps of the projected key points and offset vectors for the 8 projected corner points of the 3D bounding box, among others. These contexts enhance the model's ability to interpret complex 3D environments from 2D data.

2) Training and Inference Strategy:

- **End-to-End Training**: MonoCon is trained end-to-end, optimizing the feature backbone and regression heads simultaneously. This holistic approach ensures that the model effectively learns both essential and auxiliary tasks during training.
- **Post-Training Efficiency**: After training, auxiliary context regression branches, crucial for the learning phase, are discarded. This step significantly boosts the inference efficiency of the model, a vital aspect for real-time applications like autonomous driving.
- **Loss Functions**: The model employs various loss functions to optimize its performance, including Gaussian kernel weighted focal loss for heatmaps [6], Laplacian aleatoric uncertainty loss for depth estimation [7], as well as the others. These functions are key to the model's accuracy and efficiency.

3) Implementation: The implementation of the MonoCon model in our project was guided by the framework's core principles, as detailed in the MonoCon paper, and adapted for our specific use case in the NA565 final project. The implementation steps included:

1) **Environment Setup**: Our development environment was established on Google Cloud using a Virtual Machine configured with NVIDIA T4 GPU. This VM setup was crucial for the computational demands of our project. We created a new conda environment, cloned the MonoCon repository from GitHub, and installed the necessary dependencies from the 'requirements.txt' file, ensuring a stable environment for the development and testing of the monocular 3D object detection model.
2) **Data Preparation**: As per the MonoCon method, our implementation focused on the KITTI benchmark, with image dimensions of 384×1280. We ensured that our dataset was compatible with the model's requirements, especially in terms of image size and format.
3) **Model Training**: Following the preparation of our dataset, we proceeded to train the MonoCon model. This involved using the pre-configured settings in the MonoCon PyTorch repository. We utilized a single GPU setup for training, in line with the repository's capabilities.
4) **Model Evaluation and Inference**: After training, we evaluated the model's performance using the provided scripts in the repository. For inference, we leveraged the MonoCon model's ability to compute class-specific heatmaps for 2D bounding box centers and other relevant parameters, such as the offset vectors for 3D bounding

box projection. As it is the official evaluation protocol for Kitti 3D Object detection [8], we use 3D Average Precision with an Intersection over Union threshold of 0.5, integrated over 40 uniformly distributed recall intervals. We use this as our relative evaluation metric, which allows us to evaluate our improvement and compare our results to other models.

5) **Auxiliary Context Learning**: Consistent with the MonoCon approach, our implementation also focused on learning auxiliary contexts, including attentive normalization and channel-wise feature attention, to enhance the model's ability to interpret 3D environments from 2D data.

III. Results

As a first attempt, the neural network was trained for 200 epochs with all other hyperparameters carried over from the MonoCon Repo. We were able to pass all of the normal tests with the results of this training. This also resulted in an AP score of 4.655 for the extra challenge set. We subsequently ran the neural network for a number of epochs and the AP scores for the extra challenge set are tabulated below:

Number of Epochs	AP
300	9.6555
350	7.1766
450	7.8824
500	8.8371

TABLE I
AP Scores for Extra Set at Various Epochs

The AP score of 9.6555, with 300 epochs, was submitted to the leaderboard.

Fig. 2. Example Network Output

Figure 2 shows a visualization of one of the outputs from the model - a 3-dimensional bounding box around the car.

IV. Discussion

Overall, the results above show that this architecture is promising for its functionality and capability for transfer learning. Once the environment setup was done, we achieved good performance training on about 1000 images in about 7 hours, for a few dollars worth of compute using a mid-tier GPU. Teams working on solutions for a production environment, have access to orders of magnitude more data and compute power. From I, in our relatively simple environment, the model started to over-fit the training data in a relatively short amount of time. To see better performance and reliability, it would be necessary to use additional data, augment the existing data, or add more complexity to the model. Figure 1 and 2 show that the image set is quite realistic for a simulation and valuable training and evaluation set, it seems to lack some of the noise and details that would exist in a real-world environment with real-world camera sensors. So there are opportunities to expand the model and training pipeline that would require longer and more intensive training but would produce a higher fidelity model that is more robust in actual driving scenarios.

V. Conclusion

The training and testing we did showed that the architecture of the MonoCon detector provides a flexible foundation for transfer learning and application to a new environment. With a reasonably small number of epochs training on the new data set, the architecture provided impressive results on an inherently challenging task - extracting 3D pose information from a 2D image. A potential strategy for improving the model's success on the challenge data set could include augmenting the training data to add some noise, or fog-like characteristics, or varying the overall illumination in the images. This could provide additional and novel data to train the model. Another architectural change that could improve performance in changing conditions could be converting the image data into a colorspace like Hue Saturation Lightness(HSL) instead of RGB because this separates some of the variations due to changes in the illumination of the scene and would allow the network to learn more pointedly based on the features in the Hue and Saturation channels. Additionally, to fully understand this architecture's portability and reliability, it requires testing on different real-world data sets. Our testing only used images from a simulation, which is useful for providing a proof of concept. It simply needs to be tested with more data, and data more representative of the real world.

References

[1] P. Salmane, J. Rivera Velázquez, L. Khoudour, N. Mai, P. Duthon, A. Crouzil, G. Pierre, and S. Velastin, "3d object detection for self-driving cars using video and lidar: An ablation study," *Sensors*, vol. 23, no. 6, p. 3223, Mar 2023.

[2] T. Wang, X. Zhu, J. Pang, and D. Lin, "FCOS3D: fully convolutional one-stage monocular 3d object detection," *CoRR*, vol. abs/2104.10956, 2021. [Online]. Available: https://arxiv.org/abs/2104.10956

[3] S. Srivastava, F. Jurie, and G. Sharma, "Learning 2d to 3d lifting for object detection in 3d for autonomous vehicles," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4504–4511.

[4] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[5] X. Liu, N. Xue, and T. Wu, "Learning auxiliary monocular contexts helps monocular 3d object detection," *CoRR*, vol. abs/2112.04628, 2021. [Online]. Available: https://arxiv.org/abs/2112.04628

[6] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *CoRR*, vol. abs/1708.02002, 2017. [Online]. Available: http://arxiv.org/abs/1708.02002

[7] Y. Chen, L. Tai, K. Sun, and M. Li, "Monopair: Monocular 3d object detection using pairwise spatial relationships," *CoRR*, vol. abs/2003.00504, 2020. [Online]. Available: https://arxiv.org/abs/2003.00504

[8] A. Geiger, "The kitti vision benchmark suite, url = https://www.cvlibs.net/datasets/kitti/eval$_object_detail.php?result = 5e17cbbabbf775d8cc376793168be49bd6f01608, year = 2023.''