Hybrid Monocular Distance Estimation using CNN + Pinhole Residual

Mohamad Ghanem

1 Introduction

Estimating object distance from a single RGB image is a fundamental perception task in vision. Depth from a monocular camera is ambiguous in theory, but real scenes provide consistent cues such as object scale in pixels, ground contact position, and class priors. Unlike stereo vision or LiDAR, which provide direct depth measurements, monocular depth estimation is theoretically ill-posed (Shi et al., 2023). The KITTI dataset (Geiger et al., 2013) provides per-object 2D boxes and 3D locations in camera coordinates, so the forward distance Z can be used as a supervised training target. The projection of a 3D world onto a 2D plane results in an inherent loss of depth information. This report will explain the procedures taken to try and achieve a reasonable model that detects the distance from an image. This is a multi-output model that classifies the object, and detects the distance.

1.1 Motivation

There is an opportunity for innovation for this topic by using a computationally efficient solution. Existing solutions often rely on dense depth map estimation or complex 3D bounding box regression, which can be computationally intensive and treated as black-box systems. Shi et al. (2023) as well as Lee et al. (2022) explain the need of a lightweight distance estimation method, explaining that computationally expensive dense depth-estimation for real-time driving scenarios. Davydov et al. (2022) also explains the preference of pinhole camera physics that is lightweight, object-specific over dense depth maps.

1.2 Objective

The objective is to build a lightweight convolutional neural network (CNN) that predicts per-object distance $\Delta\hat{Z}$ from cropped detections and improve the physics baseline derived from the pinhole camera model. Providing a fast metric for predicting values when computer vision tasks can be computationally heavy. To get the distance, we perform the following actions:

1. The model predicts $\Delta\hat{Z}$ from the model.br
2. We calculate the pinhole formula:
$$Z_{pin} = \frac{f_x \cdot \bar{H}}{h_{pixel}} \tag{1}$$
Where f_x is the focal length from the image calibration.
And $\bar{H} = \sum p(c) \cdot H_c$ is the class probability weighted average height from training data.
And h_{pixel} is the height of object in pixels.
3. To account for the variables affecting the physicis principle in the monocular camera (i.e camera pitch, hues, colors, object placement etc.), after calculating Z_{pin}, the CNN model is employed to correct the pinhole camera.

1.3 Related Prior Work

1.3.1 A lightweight distance estimation method using pinhole camera geometry model, Han et al. (2025)

Han et. al relies purely on a geometric approach (using YOLOv8 only to find the box), whereas the model in this paper work uses a hybrid approach (using the formula as a "first guess" anchor and a CNN to learn a correction, i.e. residual learning). Han et. al rely on finding the object focal length (using the same formula in this paper) but it is re-arranged to $f_y = \frac{\cdot h_p}{h}$. Han et al. (2025) mainly relies on using the YOLO-V8 model to detect object bounding box, then using the pinhole formula to mathematical reach a solution. The model in this paper achieves the solution by correcting the residual found $\hat{Z} = Z_{pin} + \Delta\hat{Z}$. Where $\Delta\hat{Z}$ is the value found through training a CNN model.

1.3.2 Davydov et al. (2022), Supervised Object-Specific Distance Estimation from Monocular Images for Autonomous Driving

This is similar to what Davydov et al. (2022) in case of using the pinhole formula. However, they do not calculate a fixed anchor based on dataset averages. Instead, their network predicts the physical dimensions which are then fed into a hard-coded decoder layer that enforces the pinhole formula. They relied on the model to predict the object's physical size which makes it dynamic rather than statistical, the approach taken in this model is more statistical. Uses a dynamic crop factor (λ_c) relative to the image size, resizing inputs to 640×194 to preserve aspect ratio. In the dataset, it was recognized that the $75th$ percentile of the dimensions are 93×93. Therefore the model uses input size of 128×128.

1.3.3 Structured deep learning based object-specific distance estimation from a monocular image by Shi et al. (2023)

Shi et al. (2023) argue that traditional optical/mathematical models (like the Pinhole model here) are "ill-conditioned" and "require strict prior conditions" that fail on curved roads or when camera angles change. The model in this paper counters it by using the pinhole as an anchor only, not the final answer. Therefore, using a CNN (ResNet) to learn a residual $\Delta\hat{Z}$ that specifically corrects for the systematic errors (like pitch changes or road gradients) that Shi et al. (2023) mentions are what causes these methods to be "ill-conditioned" (Shi et al., 2023, p. 4152).

1.3.4 Deeper Depth Prediction with Fully Convolutional Residual Networks by Laina et al. (2016)

(Laina et al., 2016) demonstrated that using a ResNet-50 encoder allowed for a larger receptive field and deeper feature extraction without the vanishing gradient problem, significantly outperforming previous state-of-the-art method. Their goal was to produce a high-resolution depth map where every pixel has a value. To achieve this, they invented Up-Projection Blocks to increase the spatial resolution of their feature maps. From Laina et. al work, it was clear that to get a reasonable distance measure, the model must learn the concept of geometry, perspective, and scale entirely from the training data pixels, which is why the extra geometric MLP is added. The loss function used by Laina et al. (2016) was also implemented in the model (instead of the berHu loss, a smooth L1 loss was used), with the addition of distance-based weighting to penalize both extremely far and close objects.

2 Methods

The approach for achieving a solution to this problem is to use a CNN (Convolutional Neural Network) as recommended by Davydov et al. (2022) and Mousavian et al. (2017). Another solution would be to use pure deep learning (Han et al., 2025). CNN uses filters to extract features from images, such as edges, textures, and patterns, by applying kernel filters across the input. The model has fully connected layers, and since this is a multi-output model, there are two main heads, for regression and classification. There are two main components to the model:

2.1 Model Architecture

To experiment, two types of model architectures were investigated, once with a pre-trained model with the last head removed and replaced with the model's prediction heads, and another with a custom 4 distinct CNN blocks.

2.1.1 4 Distinct blocks model - Custom CNN

To establish a performance baseline, a lightweight CNN was designed and trained from scratch. The architecture follows a design pattern consisting of four convolutional blocks. The input is a Input: 128×128 RGB image crop, where the feature channel increases in sequence ($32 \to 64 \to 128 \to 256$), while the spatial dimension decreases each block ($128 \to \ldots 8$). The final output is a feature map size of $256 \cdot 8 \cdot 8$ that is flattened.
Here is what each block does:

1. Convolutional layer with a 3×3 kernel to detect patterns (shapes, edges etc.) in the image while keeping the image size the same (padded)

2. Use a batch normalizer to stablize the gradient
3. *ReLU* (Rectified Linear Unit) for the activation function to introduce non-linearity in the model
4. Pooling, to keep the highest value (most important features) in the image.

Table 1: Custom 4-Block CNN Architecture. Each block consists of: Conv2d (3×3) + BN + ReLU $\rightarrow$ MaxPool (2×2).

Stage	**Channels** ($In \rightarrow Out$)	**Spatial Dim**	**Output Size**
Input	3 (RGB)	128×128	-
Block 1	$3 \rightarrow 32$	64×64	$32 \times 64 \times 64$
Block 2	$32 \rightarrow 64$	32×32	$64 \times 32 \times 32$
Block 3	$64 \rightarrow 128$	16×16	$128 \times 16 \times 16$
Block 4	$128 \rightarrow 256$	8×8	$256 \times 8 \times 8$
Flatten	$256 \rightarrow 16,384$	1×16384	16,384

2.1.2 ResNet-18 CNN Model

For the primary model, a ResNet-18 backbone is utilized pre-trained on the ImageNet dataset. The final layer fully connected classification layer is removed and the network is used as pure feature extractoin. This leverages millions of natural images, which are critical for monocular depth cues. The output is 512 feature representation (mathematical representation of the image). Laina et. al states that using ResNet-50 is better than other models for task prediction (specifically VGG and AlexNet), however, ResNet-50 is way more power than what is needed here, so it is decided to settle with ResNet-18. (Laina et al., 2016)

2.2 Prediction Heads

In both backbones, the visual features are concatenated with the 32-dimensional geometric feature representation from the bounding box MLP. The geometric MLP encodes the normalized bounding-box parameters $[x_{norm}, y_{norm}, w_{norm}, h_{norm}]$ into a 32-dimensional feature vector. These features restore spatial context lost when cropping and resizing the image, allowing the network to learn depth-relevant cues such as object position in the frame, scale, and aspect ratio.

This fused vector is then passed to two parallel fully connected heads (one for classification, one for regression), each containing a hidden layer of size 512 with ReLU activation and Dropout ($p = 0.5$) to prevent overfitting. These values were found through experimentation.

2.3 Dataset

For the dataset, the KITTI Object Detection Benchmark is used, a standard dataset for autonomous driving research Geiger et al. (2013). The dataset consists of synchronized image and LiDAR data captured from a moving vehicle in urban environments. The images folder contain high resolution RGB images (approx. 1242×375 pixels). Label text files containing 2D bounding boxes $(x_{min}, y_{min}, x_{max}, y_{max})$, the object classes (e.g., Car, Pedestrian), and the ground truth 3D location (x, y, z). The calibration text file contains camera intrinsic matrices ($P2$) required to calculate focal lengths for geometric estimation. For each object in the image, the object is cropped with a 10% padding around the box (Davydov et al., 2022; Shi et al., 2023), , the object image is then resized into 128×128 for the input of the model. To improve the results of the model, the co-ordinates of the 2D bounding box is also stored along with the width and height of the image, which are normalized during training.
The dataset provides the ground truth value Z_{gt}. The average physical height (H_{avg}) for each class (e.g., Car $\approx 1.53m$) by iterating through the entire training split during initialization. This statistic serves as the critical prior for our Pinhole Geometry formula. As an important note, training and testing sets are properly split, making sure that objects obtained from a training image set are included in the testing set (Strictly separating the training and testing objects). That way, when infering an image, it is an completely based on an unseen image.

2.3.1 Data splitting

The data is split into an official training set and a validation set. We ensure that crops from the same original image do not leak between sets to maintain a strict independence between training and testing environments.

2.3.2 Data augmentation

To improve model robustness and prevent overfitting, random augmentations is applied during training. By randomly adjusting brightness, contrast, and saturation, a color jitter is introduced. Additionally, each image crop is randomly flipped (while keeping the distance label unchanged), other techniques involve inversion instead of flipping like Tang and Chen (2019) or mirroring by Mousavian et al. (2017). All images are normalized using the standard ImageNet mean and standard deviation ($\mu = [0.485, ...], \sigma = [0.229, ...]$) (set values of of the ResNet-18 model statistics) to align with the pre-trained weights of the ResNet backbone. Similarly, for the custom 4 block CNN, the sttandard deviation and the mean of the training dataset are calculated when normalizing to be accurate. These techniques are essential to prevent the model from overfitting, which are methods also used by Davydov et al. (2022) (but strong jitter for specific minority classes their case), and increase the ability to predict distances from images regardless of their position (left, right, centered etc.) or the color (by adding color jitter).

2.4 Validation Strategies

For splitting the data, a typical 80% training to 20% testing split was done. More importantly, it was ensured that crops originating from the same source frame were not split across sets to prevent data leakage, where the model might memorize the specific lighting or background of a scene.

2.4.1 Metric Evaluation

To determine the performance of a model, it was using four key metrics, prioritizing safety-critical accuracy.

Since there are two heads, two loss functions were used:

1. Classification Loss: Cross Entropy Loss

$$L_{cls} = -\log(\frac{\exp(x_y)}{\sum_{c=1}^{C} \exp(x_c)}) \tag{2}$$

2. Regression Loss: Smooth L1 Loss

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^T \tag{3}$$

$$l_n = \begin{cases} 0.5(x_n - y_n)^2/\beta, & \text{if } |x_n - y_n| < \beta \\ |x_n - y_n| - 0.5 \cdot \beta, & \text{otherwise} \end{cases} \tag{4}$$

Where $\beta = 1.0$.

Finally, to account for the safety-critical nature of near objects, the regression loss is weighted by the inverse of the ground truth distance:

$$L_{reg} = \frac{1}{Z_{gt} + 1} \cdot \text{Smooth}L_1(Z_{pred}, Z_{gt}) \tag{5}$$

To combine these two distinct objectives into a single training signal, a weighted multi-task loss function is employed (Tang and Chen, 2019):

$$L_{total} = L_{cls} + \lambda \cdot L_{reg} \tag{6}$$

Where λ is a hyperparameter that controls the task priority. Since the ResNet backbone is pre-trained on ImageNet and rapidly achieves high classification accuracy ($> 95\%$), the model initially struggles primarily with the geometric regression task. Therefore, λ is used to scale up the regression gradients, ensuring the optimizer prioritizes minimizing distance errors over refining an already accurate classification. This idea was implemented from the Micrsoft research for the implementation of *Fast R-CNN* by Girshick (2015).

Other metrics calculated for further evaluation, specifically to see the performance of the regression head (distance estimation), these metrics were used by Davydov et al. (2022), Shi et al. (2023), and Han et al. (2025)

1. Mean Absolute Error (MAE): The primary metric, measuring the average absolute difference in meters between Z_{pred} and Z_{gt}.
2. Root Mean Squared Error (RMSE): Used to identify large outliers. Since RMSE penalizes large errors heavily, a low RMSE indicates the model is robust against catastrophic failures (e.g., predicting 50m when the object is 10m away).
3. Classification Accuracy: The percentage of correctly identified object types (Car vs. Pedestrian vs. Cyclist), although when using ResNet-18, this was not given that much importance as that backbone is pre-trained with weights.
4. Nearest Object Accuracy: A key novelty in this paper is to introduce a safety-specific metric that measures how often the model correctly identifies which object in the scene is physically closest to the object.

3 Results

The results from the experiments were implemented to build a supermodel, the following configurations were chosen for both the ResNet-18 and the custom 4 block CNN.

3.1 Quantitative Results

Out of 21 experiments conducted, below are some of the most influential changes that have been done

3.1.1 Results from Experimentation and Hyperparameter tuning

Table 2: Hyperparameter Tuning - over 5 epochs

Parameter	**Values Tested**	**Baseline**	**Impact Area**
Backbone	Custom (4-Layer), ResNet-18	ResNet-18	Feature Extraction Quality
Learning Rate	1e-3, 5e-4, 1e-4	0.001	Convergence Stability
Lambda Reg (λ)	1.0, 5.0, 10.0	1.0	Task Balancing (Class vs. Distance)
Batch Size	16, 32, 64	32	Gradient Noise / Regularization
Dropout	0.3, 0.5	0.5	Avoid overfitting
Hidden Dimensions (Neurons)	256, 512, 1024	512	Number of neurons
Weight Decay	1e-4, 1e-3, 0.0	0.0001	Penalizing large weights (Overfitting)
Step Size	5,10	7	Period of learning rate decay

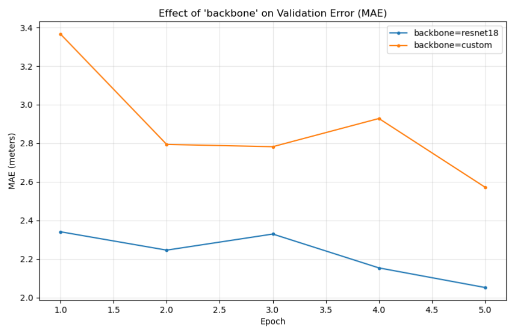


Figure 1: MAE Backbone comparision

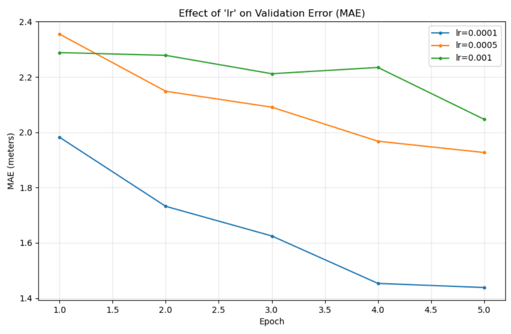
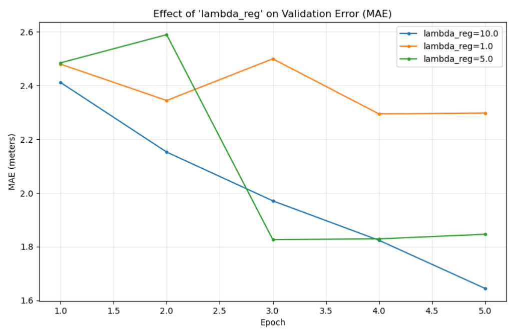


Figure 2: MAE Learning Rate comparision

Although these results did not introudce any drastic difference, they were important with producing an accurate distance estimate. A higher learning accelerates the initial learning process but increases the risk of instability, a low learning rate here ensures stable, cautious convergence but can make the training process slow or achieve suboptimal solution.



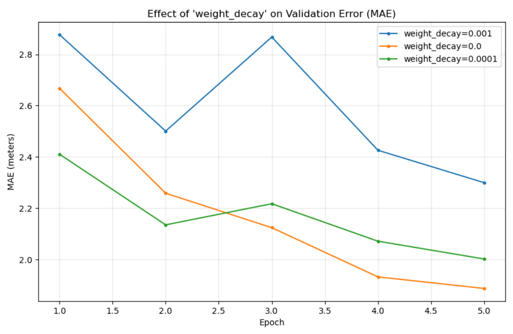


Figure 3: MAE Lambda Regularization comparision

Figure 4: MAE Weight Decay comparision

A lambda that is high (e.g. here 10.0 is used), will cause the model to prioritize optimizing the distance prediction (which is the intention here). The gradients from the regression task will dominate the overall training signal. $\lambda < 1$ (e.g., 0.1) prioritizes optimizing the object classification, and lambda is set to 1.0 will treat both predictions with equal initial importance.

For weight decay, a high weight decay value indicates a strong regularization penalty, which attempts to reduce overfitting but risk underfitting (basically the bias to variance tradeoff), this typically produces a smooth output. While with a low weight decay, the optimizer is given much more freedom to increase the weights to fit the training data. The L2 penalty has a minor influence on the total loss, thus increasing overfitting risk.

The results suggest that weight decay be set to 0.0 (or off), or a better alternative would be to use the AdamW optimizer which results in more effective regularization and better generalization.
There are other experiments that were done, but the results were not too significant to include in the report.

3.1.2 ResNet-18 Backbone Supermodel

Final Configuration: Backbone: ResNet-18 · Learning Rate: 1e-4 · Lambda Reg (λ): 10 · Batch Size: 64 · Dropout: 0.5 · Hidden Dim: 512 · Weight Decay: 0.0 · Step Size: 10.

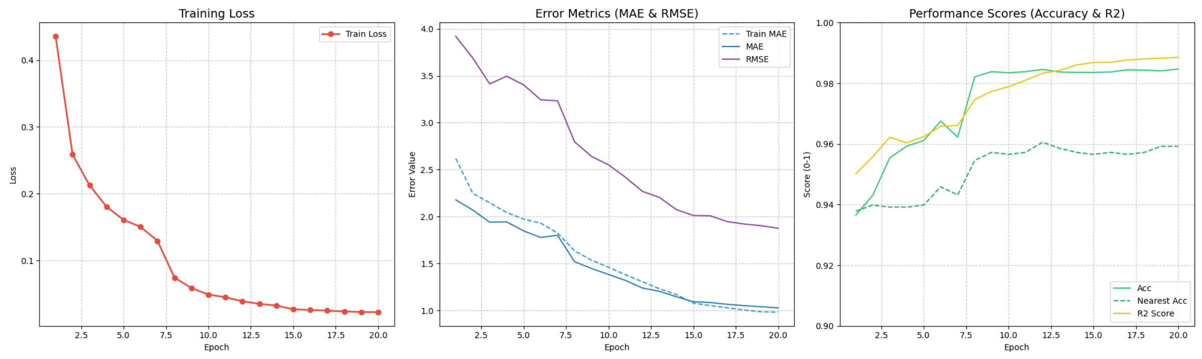


Figure 5: Evaluation Metrics for the Supermodel
center

3.1.3 Training Loss - ResNet-18 Model

The training loss curve exhibits a textbook "L-shaped" convergence, characterized by a rapid initial descent followed by a plateau. The sharp drop in loss around epoch 8 is the most critical feature, By epoch 15, the curve effectively flatlines near 0.02, confirming that the model has fully converged and squeezed all possible learning from the dataset, making further training redundant.

3.1.4 MAE & RMSE Error Metrics - ResNet-18 Model

The error metrics indicate a highly precise model, achieving a final Validation Mean Absolute Error (MAE) of approximately 1.03 meters, meaning the average prediction is within a meter of the actual distance. However, the distinct gap between the MAE and the higher Root Mean Squared Error (RMSE $\approx$ 1.88m) reveals the model's sensitivity to distance; because the geometric pinhole formula is non-linear, a tiny 1-pixel error on a far-away object results in a massive depth error (e.g., predicting 90m instead of 80m). These occasional large outliers are penalized heavily by the RMSE, keeping it higher than the MAE, which is a standard characteristic of geometric depth estimation.

3.1.5 Accuracy & R^2 Performance Scores - ResNet-18 Model

The performance scores demonstrate exceptional reliability, with the R^2 score reaching a near-perfect 0.99 and classification accuracy hovering around 98%. These high scores validate the Hybrid archtectural approach, where the neural network is succesfully handling the easy part (classifying whether the object is a Car or Truck), which allows the hard coded geometric logic to handle the calculating depth. Since the network rarely misclasifies objects, the resulting depth correlation remains extremely high and stable.

3.1.6 Quantitative Results - Custom 4 Block CNN

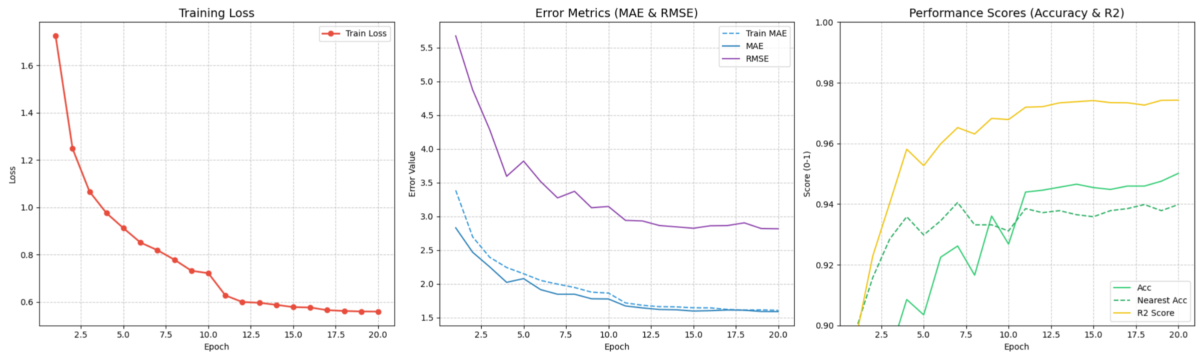


Figure 6: Evaluation Metrics for the Supermodel of the Custom CNN model center

3.1.7 Training Loss - Custom 4 Block CNN

The training loss curve demonstrates a steady but slower convergence compared to the ResNet supermodel, starting at a higher initial value of approximately 1.73 (which is expected, this is not pre-trained). While the model successfully learns the task, reducing the loss to roughly 0.56 by epoch 20, it lacks the sharp "L-shaped" drop seen in the pre-trained backbone. The curve does not fully flatline near zero, suggesting that this shallower architecture is still struggling to fit the complex features of the dataset and has likely hit its capacity limit, unable to extract the finer nuances required for higher precision.

3.1.8 MAE & RMSE Error Metrics - Custom 4 Block CNN

The error metrics quantify the limitations of the custom backbone, yielding a final mean absolute error (MAE) of approximately 1.59 meters. This represents a roughly 54% increase in error compared to the ResNet model (1.03m), proving that the custom network is under-parameterized for this specific regression task. The significant gap between the MAE and the Root Mean Squared Error (RMSE $\approx$ 2.82m) persists here as well, indicating that without deep feature extraction, the model is highly susceptible to severe outliers, particularly on distant objects or very close objects where a lack of visual detail leads to large penalties in the geometric calculation.

3.1.9 Accuracy & R^2 Performance Scores - Custom 4 Block CNN

Despite the reduced capacity of the network, the performance scores remain relatively robust, with classification accuracy reaching 95% and an R^2 score of 0.97. This result reinforces the validity of the Hybrid architecture design: even with a weaker feature extractor, the geometric priors act as a safety net, ensuring that the general distance correlation remains high. The Nearest Object Accuracy remains strong at $\approx 94\%$, demonstrating that while the absolute distance values are less precise than the supermodel (using ResNet-18), the custom model remains functionally useful for identifying close proximity in safety-critical contexts. If improved, perhaps with more layers or neurons, it could show close results to the ResNet-18 model.

3.2 Qualitative Results / Inference on unseen images

For these inference images, a pre-trained Faster R-CNN (a lightweight model) was used to generate the initial 2D bounding boxes, which were then cropped and passed to the Hybrid Distance Model for depth estimation and classification.

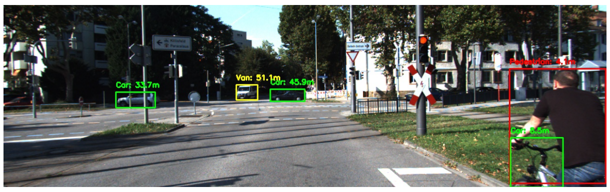


Figure 7: Inference id: 007501
Classification and Regression errors can be noticed here, due to the object cut out and the limitation of object tracking (i.e the pre-trained weights of ResNet-18 do not know what a Cyclist is, and the model identified the cycle as a car, possibly due to lack of informaiton in th crop)

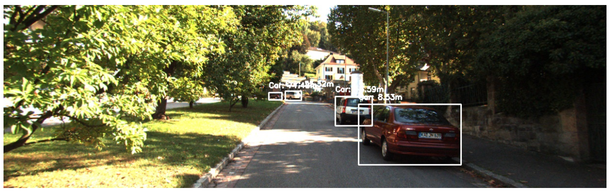


Figure 8: Inference id: 0000044
Great results achieved in this image inference, results look realistic and accurate

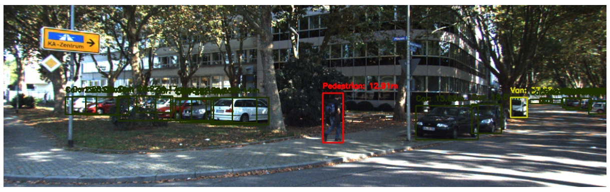


Figure 9: Inference id: 001060
Predictions look somewhat accuracte, all objects classified correctly, distances look somewhat accurate by eye

4 Discussion

While the Hybrid Monocular Model successfully demonstrates the viability of combining geometric priors with deep learning, several limitations were identified during experimentation that constrain its real-world applicability.

4.1 Limitations and Future Work

4.1.1 Sensitivity to close and far objects

A primary limitation to the geometric approach is the non-linear sensitivity to 2D bounding box errors at long distances. As observed in the quantitative results, the gap between the MAE ($1.03m$) and RMSE ($1.88m$) indicates that the model is prone to large outliers. Mathematically, the pinhole formula relies on the inverse of the 2D box height ($Z \propto 1/h_{pixel}$). For distant objects, h_{pixel} is very small. Thus, a deviation of merely 1 or 2 pixels in detection results in a depth error of several meters. Future work should move beyond simple 2D inputs and implement 3D Bounding Box Regression, similar to the method proposed by Mousavian et al. (2017). By training the network to predict the 3D dimensions (height, width, length) and orientation of the object, the system could enforce stricter geometric consistency constraints (ensuring the 3D box projects correctly into the 2D image) rather than relying on a single, noisy 2D height measurement. Another possiblity is that if the relative errors are penalized ($\frac{|Z_{pred}-Z_{gt}|}{Z_{gt}}$) rather than absolute errors, this could potentially stabilize predictions for far-away objects (i.e. Car at 50m (pred 55m): The error is $5/50 = \mathbf{10}\%$, would be a good guess, however car at 5m (pred 10m): The error is $5/5 = \mathbf{100}\%$ would be catastrophic.)

4.1.2 Class-Specific Ambiguities

Qualitative inference revealed that the model struggles with class-specific geometry when the pre-trained backbone lacks domain-specific knowledge (see figure 7). Specifically, the model failed to accurately track the "Cyclist" class. This is likely because the ResNet-18 backbone is pre-trained on ImageNet, recognizes generic "people" and "bikes" but does not understand the structure of a "Cyclist" as defined in the KITTI dataset. This misalignment leads to incorrect height priors (H_{avg}). Future work involves fine-tuning the entire backbone on the KITTI dataset rather than using it purely as a frozen feature extractor, allowing the model to learn distinct visual features for ambiguous classes. Future work should involve deployment on embedded edge devices similar to the experimental setup demonstrated by Lee et al. (2022), to evaluate inference latency in real-time traffic scenarios, **although this requires more resources**, see next point.

4.2 Different lighting variations, and camera motion

To start, day and night will certainly make a big difference as the model is primarily contains only day-time, dry, and sunny images Geiger et al. (2013), making inference on night time objects irrelevent (it is not possible include any random images as it violates i.i.d assumptions), therefore, there is a need to collect more distributed data to capture all scenarios (snow, rain, desert climates, etc.) Lee et al. (2022) developed the assitance system noticed variations between day/night accuracies. Han et al. (2025) also addressed these concerns as next steps for real world integration systems. As a more advanced future work, the model can be enhanced by using ConvNeXt for inference, which was also used byDavydov et al. (2022).

4.3 Implications of the Work

Despite these limitations, this project has significant implications for the field of autonomous perception, particularly regarding efficiency and explainability.

4.3.1 Efficiency in Embedded Systems

The most significant implication is the demonstration that lightweight, object-level distance estimation is feasible without heavy dense depth mapping (Davydov et al., 2022; Shi et al., 2023) . Existing solutions often require computing a depth value for every pixel in a generic scene, which is computationally expensive. By restricting the problem to specific objects of interest (Cars, Pedestrians) and leveraging a simple mathematical prior, our model achieves high correlation ($R^2 \approx 0.99$), with a fraction of the computational cost. This suggests that such hybrid architectures are highly suitable for resource constrained embedded systems, such as delivery robots or dashcams, where full LiDAR or stereo-vision setups are cost-prohibitive.

5 Conclusion

This paper has discussed the effects of using a physics anchor (like a heuristic) and train a CNN model for classification and residual learning to estimate distances from images, where the system overcame the theoretical ill-posed view of monocular depth estimation without relying on computationally expensive dense depth maps. The ResNet-18 Supermodel demonstrated exceptional precision, achieving a Mean Absolute Error (MAE) of 1.03 meters and an R^2 correlation of 0.99. This significantly outperformed the custom 4-block CNN baseline ($MAE \approx 1.58m$ and $R^2 \approx 0.94$). Although there is a huge room for improvement for this project especially with MAE and RMSE revealing limitations in extremely long/short or 'blind spot' range geometric estimation, it has demonstrated that a physics informed NN (residual learning) is an option for lightweight and computationally efficient detection models with room for improvement.

References

Davydov, Y., Chen, W.-H., and Lin, Y.-C. (2022). Supervised object-specific distance estimation from monocular images for autonomous driving. *Sensors*, 22(22):8846.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231–1237.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448.

Han, T.-T., Nguyen, A.-T., Nguyen, M.-C., Phung, V.-H., Nguyen, C.-P., and Ngo, B.-V. (2025). A lightweight distance estimation method using pinhole camera geometry model. *Measurement Science and Technology*, 36(4):045406.

Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., and Navab, N. (2016). Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 239–248.

Lee, K.-F., Chen, X.-Z., Yu, C.-W., Chin, K.-Y., Wang, Y.-C., Hsiao, C.-Y., and Chen, Y.-L. (2022). An intelligent driving assistance system based on lightweight deep learning models. *IEEE Access*, 10:111888–111900.

Mousavian, A., Anguelov, D., Flynn, J., and Košecká, J. (2017). 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7074–7082.

Shi, Y., Lin, T., Chen, B., Wang, R., and Zhang, Y. (2023). Structured deep learning based object-specific distance estimation from a monocular image. *International Journal of Machine Learning and Cybernetics*, 14(12):4151–4161.

Tang, X. and Chen, L. (2019). An unsupervised monocular image depth prediction algorithm based on multiple loss deep learning. *IEEE Access*, 7:162405–162414.

6 Github Repository

https://github.com/2025F-COMP3106/project-195.git