Inter-Vehicular Depth and Velocity Estimation using a Monocular Camera

Saharsh Agarwal*
saharsh2@andrew.cmu.edu

Siddhartha Namburu*
snamburu@andrew.cmu.edu

Vishnu Mani Hema*
vmanihem@andrew.cmu.edu

Ninaad Damis*
ndamis@andrew.cmu.edu

Prof. Bhiksha Raj*
bhiksha@cs.cmu.edu

Abstract

Velocity and depth estimation from cameras has been an extensive area of research in Computer Vision. With the emergence of autonomous driving, it is understood to be a complicated yet imperative task which is still being explored both in industries and academia. The goal is to know the relative velocity and depth of a specific vehicle from a monocular camera mounted on our vehicle (referred to as ego-vehicle hereafter). In this project, we aim to leverage Neural Networks and Transformers to accomplish this task. This would also help cars respond better to unforeseeable triggers from the surroundings in comparison to humans. The code for this project is available at git code.

1 Introduction

Advanced Driver Assistance Systems (ADAS) are intelligent systems designed to assist the driver in extensive ways - better reaction times, compensate for negligence and avoid mishaps. These systems provide vital information about the traffic, blockage and potential collision. Inter-vehicle distance and relative velocity with respect to self or ego-vehicle are the two most essential information for ADAS (1). With the increase in enthusiasm around driver-less vehicles - research to find the above two information has also increased significantly.

One among the representative solutions for these assistive systems is the use of fusion of sensors like LiDARs, RADARs and cameras. LiDARs are good at estimating the position and velocities of other vehicles in ideal environments. However, the data is very expensive to collect, clean and update. It also requires heavy compute for autonomous driving, making it less scalable compared to cameras. In unpleasant weather, like rain, snow or fog, or even sudden minor aberrations in the surroundings, the LiDAR's performance decreases significantly, leading to undesirable outputs. RADARs have even better accuracy, however, they suffer the same problems as LiDARs. It might give abrupt reading at crucial times, begging the question that in case the fused sensors give contradicting readings individually - whom do we listen to ?

Deep learning has provided great alternatives in many visual applications. Distance regression problems, which include depth estimation and 3D bounding boxes, have been widely studied. The relative velocity estimation or the $sceneflow$ is considered as the 3D motion comprising the distance regression problem of other vehicles relative to the ego-vehicle. Stereo cameras are commonly used for depth estimation but studies show stereo vision based object scene flow estimation in videos suffer from high computational cost. Therefore, for real time scenarios, monocular camera depth estimation or velocity estimation has been proposed as the viable alternative.

*Carnegie Mellon University, Pittsburgh, PA 15213, USA

In this work, we adopt a few recent deep learning baselines for depth and motion estimation which majorly consists of two stages: (1) **Distance regression** and (2) **Optical flow** estimation. The current baselines have dependency of bounding boxes on Distance regression and Optical flow on Velocity estimation. A robust deep depth estimator and a non-learning based method will be used to estimate velocity. Additionally, both depth and velocity will concurrently be estimated using transformers and provide real time inference for these tasks. We leverage the study towards monocular cameras using the TuSimple and KITTI Dataset.

2 Related Work

The velocity estimation primarily focuses on stacking concepts involving optical tracking, optical flow estimation and distance regression, and combining these under a single umbrella using deep neural architectures.

Object tracking is one of the fundamental problems in computer vision and has been deployed for various applications. The Lucas-Kanade method (2) uses brightness constancy assumption to track certain points of interest in images through time. Median Flow (4) extends the Lucas-Kanade method, using the concept of forward – backward error. Scene flow (5) represents the 3D motion of each point in the image. A more conventional method (6) decomposes the scene into piece-wise rigid motion planes and solves an optimization problem.

Optical flow estimation is identified as essential for efficient object tracking. Very large CNN based approaches like Flownet (7) and Flownet2 (8) stack multiple networks and utilise the warping operation. A faster network PWC-Net (9) applies the idea of spatial pyramids to Flownet2, using cost volume efficiently.

Two types of deep learning based were developed for solving inter-vehicle depth estimation, monocular depth estimation and 3D object detection. A U-Net(13) based architecture is initially proposed predicting dense depth using the concept of Conditional Random Fields. DORN(14) discretizes the depth instead and solves it using efficient learning. M3D-RPN(15) constructs a 3D region proposal network generating 3D bounding boxes utilizing features generated from 2D space. A more recent paper (16) estimates depth using spatial and temporal information in a video.

Velocity is also estimated by leveraging synthetic data in (17). A very recent approach (18) introduces MSANet to predict vehicle velocity and inter-vehicle distance estimation. (10) regresses velocity in autonomous driving settings through the trajectory features extracted from Monodepth, Median Flow tracker and Flownet. A multi-scale plane fitting based visual flow algorithm that is robust to aperture problem is described in (11). While many of these networks use different backbones to predict optical flow and velocity, (12) uses a single network to leverage the information of depth from predicting optical flow. Although sensor fusion is used in today's autonomous vehicles to estimate depth and velocity, they give more weight to cameras. Our aim is to use a single network to compute depth and velocity only using images which makes it computationally efficient in real time.

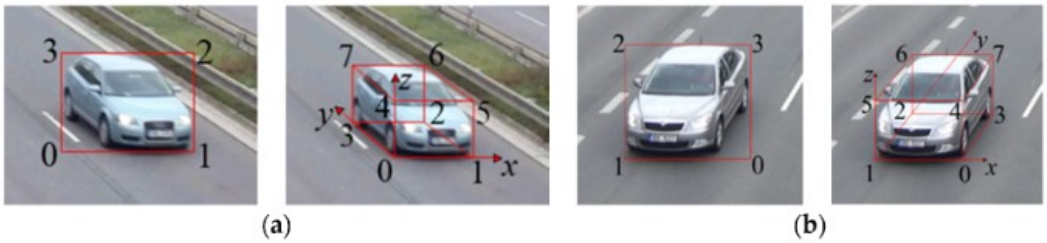


Figure 1: 2D Bounding Box and 3D bounding box in 2 different perspectives

The above approaches regress for depth using the 2D bounding box or 3D bounding projection on the image, see Fig.1. As indicated in the paper by Liu et al. (22), these keypoints have no real contextual meaning and their 2D locations vary differently with the changing of the camera view-point or object's orientation. The keypoints may lie ground, sky or trees - thereby making it difficult for the network to distinguish between keypoints and other image pixels. Hence, we use keypoints on the car itself such that their projections and their averaged center also lie on the car.

3 Dataset Processing

We will conduct experiments on TuSimple's velocity dataset (10) and KITTI 3D object detection & tracking dataset (20) for evaluating the performance of our proposed method.

KITTI: Depth is predicted on the 2017 KITTI 3D Object Detection dataset was divided into 3707 training and 3769 validation images, each displaying atleast 1 car. The dataset contains object annotations in form of 3D bounding box. The training dataset has over 14000 cars with an average length of 3.88 m, height of 1.52 m and width of 1.62 m. Velocity is predicted on the KITTI tracking dataset has 21 training sequences and 29 validation sequences benchmarked for 8 classes. We consider ony fully visible (non-truncated) **cars**.

TuSimple Velocity Challenge Dataset: This dataset provides a set of 1074 2-second-long video clips. Video sequences are captured at 20fps and thus, each is 40 frames long. Designated vehicles, at a relative distance ranging from 5 meters to up to 90 meters, are annotated with bounding boxes on the last frame along with their ground truth velocity and position generated by range sensors.

Since for a given vehicle the bounding box is defined only in the last frame, tracking over the temporal extent of the input is required for all further processing. We deploy multiple trackers and finally settle upon fast and lightweight $MedianFlow$ tracker. It precisely localizes the object outline and operates at the pixel level. The tracker provides a tight bounding box over the entire trajectory in the video, shown in Fig. 2, which is crucial for estimating the relative velocity of the objects. The tracker is initialized and run from the last frame to the first frame. This is separately done for each vehicle if multiple are present in the same video.

Figure 2: The first image is the ground truth, i.e., bounding box on the vehicle from the last frame. The next 2 images are random frames between frame numbers 1-39. The 2 frames are generated using the Median Flow trackers based on frame number 40.

4 Problem Statement

Given a video with T frames, containing i^T images and δt interval between consecutive frames, our goal is to estimate the distances and velocity in any frame i^t relative to camera coordinate system. Cars in any frame i^t are to be detected. A number of 3D points defining the car in the 3D world corresponding to the camera are directly regressed. We use 3 sets of (x,y,z) points for this - 8, 22 and 89 points for the ablation study. Inter vehicular velocity v^t are estimated leveraging the output 3D bounding box information as basis.

5 Prediction of Depth and Velocity

5.1 Monovelocity:

The baseline, shown in Fig. 3, proposes a solution by end-to-end training. The key novelty is the integration of multiple visual clues provided by any two time frames, which includes geometric clues and optical flow clues. The baseline also proposes a vehicle-centric sampling mechanism which is a pre-processing step to alleviate the distortion in the motion field. The network primarily consists of three parts as show below:

- Velocity Centric Mechanism: After bounding box is detected on the current frame, it is cropped and resized to the same input size. The same process is followed for the previous frame, which is considered in the calculations.

- The vehicle characteristics can be extracted from the features (f_i) of deep networks consisting of the PWC encoder, ROI Align and two convolution layers. It is substituted in the equation 1 below, using bottom (b), top (t), right (r) and l (left) pixel positions.

$$d_i = \frac{f_y \times d_i^h}{b_i - t_i} = \frac{f_x \times d_i^w}{r_i - l_i}, \qquad V_i = \frac{p_i - p_i'}{\Delta t} = \frac{1}{\Delta t}\begin{bmatrix} d_i \frac{u_i - c_x}{f_x} - d_i' \frac{u_i' - c_x}{f_x} \\ d_i \frac{v_i - c_y}{f_y} - d_i' \frac{v_i' - c_y}{f_y} \\ d_i - d_i' \end{bmatrix} \tag{1}$$

- On obtaining the depth from distance regression, as shown in equation above, the current velocity in x and y direction are calculated using the corrected depths (from current and previous frame). The bounding boxes and vehicle characteristics are obtained from deep features. c_x, c_y, f_x and f_y are defined by camera matrix while, u_i and v_i are pixel velocities in the x and y directions.

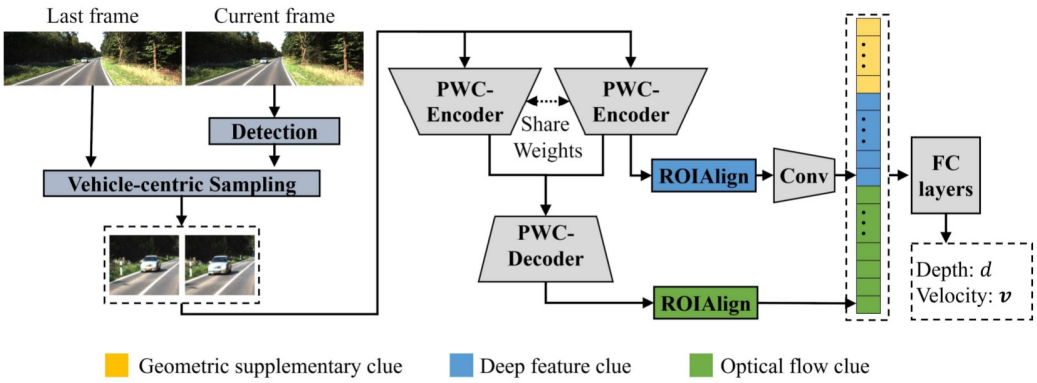


Figure 3: Network architecture proposed in Vehicle-Centric Approach (18)

5.2 Modified Monoflex for depth prediction:

A more robust depth predictor, using the ensemble method, is deployed. This approach is a 3D detection problem which involves prediction of 3D location (x, y, z), dimension (h, w, l) and orientation θ. This paper's aim is to implicitly predict dimension, orientation while directly regressing for 3D key-points. Additionally, a direct depth regressor is used to predict depth from visual cues in an image.

5.2.1 Framework Overview

The network architecture is extended from CenterNet, where objects are identified with their representative points and predicted by peaks of the heatmap. Current Monoflex has multiple regression branches which share the same backbone to regress objects properties including the 2D bounding box, dimension, orientation, keypoints and depth. The final depth is an uncertainty guided combination of directly regressed depth and a geometry based estimate from orientation, dimension and 2D key-points. The overview of the model is similar to the network shown in Fig. 4. However, after replicating Monoflex, the importance of different depth estimates are shown in Table 1. This approach predicts decoupled representation of objects i.e. dimension, orientation, depth and key-points on the 2D image. However as table 1 shows, 80% of the final depth is explained only by the depth regressor. Hence, we considered direct regression of 3D keypoints i.e. coupled representation of object instead of regressing object properties.

5.2.2 Direct regression for 3D key-points

The standard archituture of Monoflex deploys five heads after CenterNet backbone. These heads are replaced with two heads, one that directly regresses for depth and the other that regresses for 8 vertices of 3D bounding box. The coordinates of the 3D bounding box are calculated in camera

Table 1: Depth Method vs Average Precision(R40)

Depth Method	AP | R40
Ensemble	24.22
Direct depth	15.86
Key-points estimate	15.45
Direct depth + uncertainty	19.63
Key points + uncertainty	18.24

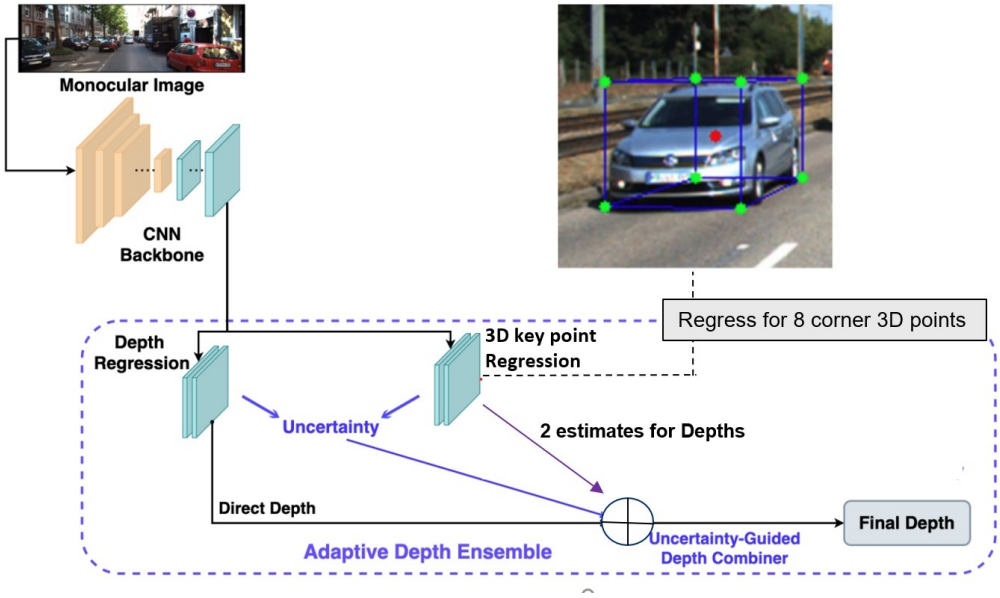


Figure 4: Depth prediction using MonoFlex

coordinate system. The aim is to implicitly learn the dimensions and orientation. After predicting the eight key-points, two estimates of depth are obtained by averaging diagonals of the 3D bounding box. The corresponding network architecture is shown in Figure 5.

5.2.3 Regress 14 more key-points

The 3d key-points corresponding to the bounding box may not have proper representation in the image since those pixels don't correspond to car, rather correspond to road/background. Hence, 14 more key-points are sampled from center at fixed dimensions from center. Currently the points are sampled using dimensions, i.e. 6 points are sampled on 6 faces of the cube and 8 more points are sampled on diagonals with a length of one fourth of diagonal. These 14 points are assumed to have more visual cues compared to the 8 key-points regressed. After the key-points are regressed two estimates of depth are obtained by averaging two diagonals of the cube. Additionally one more estimate of depth is obtained averaging the additional key-points.

5.2.4 Regress 89 more semantically meaningful key-points

After regression of 14 key-points, more semantically meaningful key-points are regressed to represent a car. These points are obtained using LiDAR data. According to the labeled 3D bounding boxes, we first segment out the individual 3D points from the entire raw point cloud. Then we remove the ground points using the ground-plane estimation method (i.e. RANSAC-based plane fitting). Finally, we obtain the "clean" 3D points for each vehicle. Corresponding points on the segmented image are calculated by projecting the points onto the image plane. The 3D points are generated by constraining the points in the LiDAR using 2D image and optimizing for reprojection loss and reconstruction loss. 3009 key-points are generated using this approach and 89 points are considered in regular intervals,

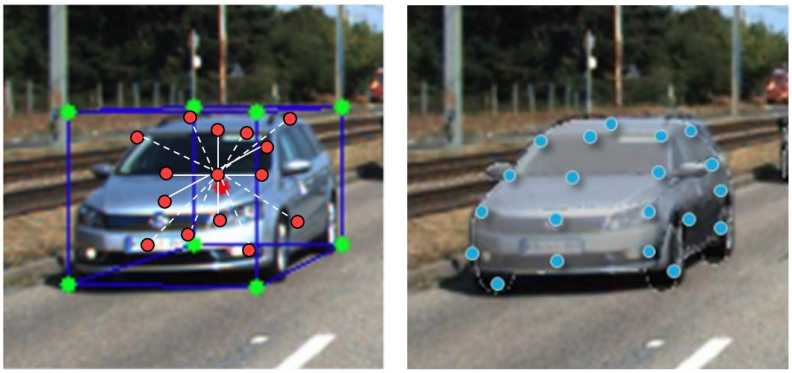


Figure 5: (a) 14 Key-points on the car. (b) 89 Semantically meaningful key-points.

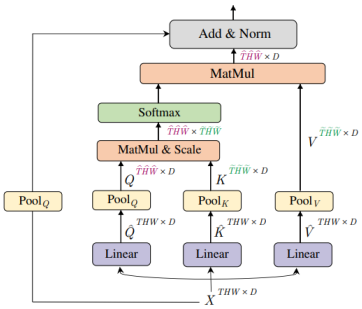
since regression for 3000 points may over fit the data. In addition to the two estimates of depths obtained from the corners, another estimate of depth is obtained by adding convolution layers. The corresponding key-points on the image are shown in Figure 5.

5.3 Removing the dependency on Optical Flow to predict velocity - AB3DMOT

After regressing the keypoints from the proposed Monoflex, a non learning based state estimation approach which is both efficient and simple in design- AB3DMOT which deploys Kalman filter constant velocity model coupled with the Hungarian method. The AB3DMOT has an additional constraint mechanism called Birth and Death Memory which limits the valid tracking and detection by considering the matched tracking only after a given threshold frame number. The velocity detected is zero until a certain threshold frame. The unmatched detections and unmatched trajectories are used to create new trajectories and delete disappeared trajectories. With the new associated trajectories the Kalman filter predicts state of trajectories and updates the trajectories, and the new trajectories are matched with 3D detections using Hungarian algorithm. The algorithm is shown in corresponding figure.

5.4 Depth and velocity regression using Multi-scale Vision Transformer

Spatio-temporal attention based blocks are used to predict depth and velocity. Leveraging the low inductive bias of a transformer, the patches in space and across time were stacked to directly regress for velocity exploiting the use of multiple multi-scale attention heads. Multiscale Transformers progressively expand the channel capacity, while pooling the resolution from input to output of the network. As shown in 6, MViT in contrast to Multi head attention, pools the sequence of latent tensors to reduce the sequence length. The pooling operator performs a pooling kernel computation along each dimensions. The pooling operator is applied to all intermediate tensors (key, query and value) independently, yielding pre-attention vectors. Attention is then computed using the shortened vectors. The pooling factor provides edge during the training period and rest of process is similar to the ViT model. Our MViT comprised of 4 scale stages, each having Multi Head Pooling Attention and MLP layers. The MViT projects the input to a channel dimensions of 96 with overlapping space time cubes of shape 3 x 7 x 7. The resulting sequence is reduced further by a factor of 4 for each additional stage and we modified the final linear layer to regress depth and velocity for every four and eight frames. The input includes the confidence score for detections and their corresponding detections and velocities. The idea behind directly regressing for depth and velocity, excluding keypoints is to make use of other non Autonomous Vehicle based - depth prediction datasets to satisfy the data hunger of transformers.



stage	operators	output sizes
data	stride 4×1×1	16×224×224
cube$_1$	3×7×7, 96 stride 2×4×4	96×8×56×56
scale$_2$	[MHPA(96) MLP(384)] ×1	96×8×56×56
scale$_3$	[MHPA(192) MLP(768)] ×2	192×8×28×28
scale$_4$	[MHPA(384) MLP(1536)] ×11	384×8×14×14
scale$_5$	[MHPA(768) MLP(3072)] ×2	768×8×7×7

Figure 6: (a) Network architecture proposed for MViT. (b) Instantiations used in the MViT.

5.5 Loss Functions

5.5.1 Monoflex

Since direct regression of 3D key-points just using a 2D image is difficult, four losses robust to scale factor variance are considered. Uncertainty is added to each of these losses. The losses investigated are as follows -

- **Inverse Sigmoid Loss**: ($z_r = \frac{1}{\sigma(z_0)} - 1$)
  Network output for directly predicted 3D coordinates has to be unlimited which is (empirically) unstable for learning. So, the output is transformed to [0,1] by sigmoid and then converted to absolute depth by using this equation, where z_o is network output. On this expression we use the L1 loss.
- Uncertainty-based L1 Loss: ($L_{dep} = \frac{|z_r - z^*|}{\sigma_{dep}} + log(\sigma_{dep})$)
  This is to minimise the uncertainty (inverse of confidence) in the multiple depths that is calculated by the network. Less confidence $\implies$ Higher σ_{dep} $\implies$ Lesser L_{dep}.
- Behru Loss: $B_L(z) = \begin{cases} \frac{z^2+L^2}{2L}, & |z| > L \\ |z|, & \text{otherwise} \end{cases}$
  It is less sensitive to outliers in data, while behaving like L1 loss for small errors and amplifying large errors.
- Scale Invariant Loss: $\frac{1}{n}\sum_i d_i^2 - \frac{1}{n^2}(\sum_i d_i)^2$
  where d is log(Predicted points) - log(Ground truth points). It helps measure the relationships between points in the scene irrespective of absolute global scale.

After testing all losses, inverse sigmoid loss is used for back-propagation of depth and uncertainty L1 loss is used for back-propagation of key-points.

6 Experiments

Implementation details for Monoflex: We adopt the same modified DLA-34 (23) as our backbone network for monoflex. All input images are padded to the same size of 384 × 1280. Every prediction head attached to the backbone consists of one 3 × 3 × 256 convolution layer, BatchNorm, ReLU, and another 1 × 1 × c_o conv layer, where c_o is the output size. The model is trained using AdamW (24) optimizer with the initial learning rate as 0.0003 and weight decay as 0.00001. We train the model for atleast 10,000 iterations with a batchsize of 8 on a single Quadro RTX 6000 GPU and the learning rate is divided by 10 at after iteration 5,000 and 8,000. The random horizontal flip with a probability of 0.5 is the only data augmentation used.

Implementation details for MViT: The backbone for this implementation is similar to MViT-B in the orginal MViT paper, with space time cube of shape 3 x 7 x 7, followed by four stages of Multi Head Pooling Attention and MLP, with a final fully connected layer to regress the confidence score, depth and velocity. The input frames was resized to 224 x 621 and cropped to 224 x 224 to satisfy the input head for MViT. Color Jitter augmentation of scale 224 x 224 was used on during training

similar to original MViT baseline. The model was trained for 400 epochs, using ADAMW with initial learning rate of 0.0001 and weight decay of 0.05 on AWS instance g4dn.4xlarge.

Evaluation Metrics:

- 3D detections are evaluated by the average precision of 3D bounding boxes (AP3D). For the validation data, we report and AP3D|R40 for a comprehensive representation, where R40 is the 40 equidistant points recall points between 0 and 1. The IoU threshold used for precision is 0.7, since this experiment considers only clearly visible cars.
- The absolute depth and velocity values are evaluated using Mean Average Error (MAE) which is standard in comparing regression values.

7 Results

7.1 Depth Regression using Monoflex

The evaluation of the proposed approach with baseline Monoflex method for 3D detection detection on the described KITTI dataset is shown in table. It can be clearly seen that using 22 key-points give the best results with not much difference in 89 key-points. So just increasing key-points may have plateaued the accuracy, instead a segmented depth map may help in increasing accuracy. Segmentation requires more image labels which are difficult to obtain. The final AP is also less than the original baseline which shows that uncoupled representation of the object could be the best way to learn the bounding box.

Table 2: Results using Monoflex approach

Model	AP | R40	Depth MAE
Baseline	24.22	0.010
Approach-1 (8 points)	19.20	0.070
Approach-2 (22 points)	21.31	0.047
Approach-3 (89 points)	21.12	0.051

Our proposed frameworks show inferior performance in comparison to the baseline Monoflex results on the KITTI Benchmark. Our most successful method utilized 22 key-points to directly regress depth, and produced a decrease in MAE of 3.7% compared to the baseline. Our least successful method used 8 bounding box points which are regressed directly.

Training time was fast as compared to the original baseline. While the baseline needs 35000 iterations to converge, our approach converges in at around 10000 iterations. Additionally, it improves the inference time by 10% since five heads are reduced to two heads.

- The 9 in appendix shows graphs how overall IoU increases for cars while training. This shows that model is able to fit the box to the car properly.
- Losses for different depth estimates are compared in Appendix. Clearly direct depth estimator works equivalently for all methods. The depth loss from corner points also converges. The depth estimates from added key-points(22 and 89) fail the predict the depth accurately (can be inferred from key-point additional MAE) and its uncertainty is high as shown in 10 and 11. The depth prediction from these key-points need to be modelled in a right way.
- Figure 10 shows that the key-point loss and key-point MAE are low which suggests that this approach is able to learn the key-points properly.
- Figure 11 shows that the key-point loss and key-point MAE are high which suggests that this approach is not able to learn the key-points properly and suggests that key-points need to be modelled in a meaningful way.

7.2 Velocity Estimation using AB3DMOT

The velocity estimated from AB3DMOT with Monoflex is compared to velocity estimated using AB3DMOT with PointRCNN. The MAE decreases by 0.03 and can be seen in the following table 3.

Table 3: Velocity MAEs

Number of Key-points	Velocity MAE
PointRCNN	0.07
Our best approach	0.10

7.3 Depth and Velocity regression using MViT

For the MViT, the ablation study was performed with the input as 4 frames and 8 frames. The framework was not able to achieve very accurate results, producing a minimum MAE of 0.61 and 0.51 for depth and velocity respectively for input with 4 frames. Figure 13 shows the training loss, validation depth and velocity MAE. The loss function converges initially for the first 1000 iterations but then it becomes stagnant for later iterations. Similar variations can be seen the depth and velocity MAE for validation.

7.4 Qualitative results

The predictions from monoflex was passed on to AB3DMOT, and the bounding boxes and velocity was visualized as shown in figure 7. In most of the cases the 3D bounding box prediction is successful and the corresponding velocity was visualized on top of the bounding box. There were few cases when the 3D bounding box visualization failed and the corresponding velocity was unable to be predicted.

The monoflex baseline was also tested on the TuSimple dataset and the 2D bounding box was obtained from the predicted 3D bounding box by considering the minimum and maximum x and y coordinate in the 3D keypoints. The figure 8.2.4 shows successful detections and failure detections. As shown in the figure the detection fails to detect cars which are far away from the ego centric vehicle but for cars which under the 15m inter- vehicle distance the detections are successful.

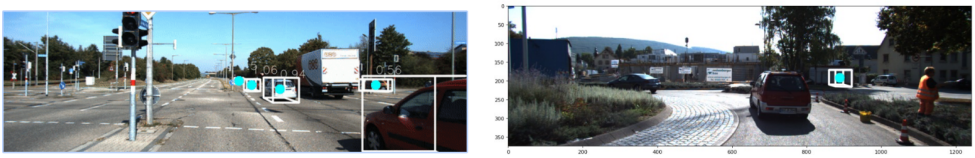


Figure 7: Monoflex and AB3DMOT: (a) Working cases. (b) Failure cases.

Figure 8: Predictions for Bounding box using MonoFlex. Predictions fail when the car is too far. Predicted distances for first five images are 16.94m, 8.72m, 9.70m and 32.47m.

7.5 Ablation Studies

Ablation Study 1: Monoflex The 3D key points regressed is hyper parameter that could be optimized. 8, 14, 22, 89 key points are regressed and the AP is compared in the following table4. Depth follows the same trend.

Table 4: Different Number of key-points

Number of Key-points	AP | R40
Baseline	24.22
8 Key-points	19.20
16 Key-points	20.11
22 Key-points	21.31
89 Key-points	21.12

Table 5: Prediction for different frame intervals

Prediction for every x frames	Depth MAE	Velocity MAE
2 frames	0.31	0.17
4 frames	0.61	0.51

Clearly performance plateaus after 22 key points. Additionally different losses described are compared and a combination of Inverse sigmoid loss and L1 loss yielded best results.

Ablation Study 2: MViT Initially L1 loss is used for regression, which yielded very bad performance. After using RMSE and SSID losses, depths and regressed for every 2 and 4 frames and the results are shown below. Clearly the model was able to predict velocities better at a 2 frame interval compared to 4 frame interval as shown in table5.

8 Conclusion

A majority of the modern approaches tackling this problem of depth and velocity estimation are very data intensive, i.e., requires a variety ground truth annotations. In this project, we have tried to reduce the dependence on some variables by implicitly learning for uncoupled representations of the object.

We showcase two approaches - **(a)** uses the monocular 3D detection architecture (Monoflex) to estimate depth, and then utilizes the Kalman Filter based AB3DMOT network to estimate the velocity of the object. **(b)** uses Multiscale Vision Transformers (MViT) consisting of spatio-temporal attention base blocks to directly regress object depth and velocity.

Experiments on the KITTI benchmark show that our first approach is able to reduce its dependence on data while experiencing just a 3.7% decrease in accuracy of depth estimation. On the other hand, our transformer based approach does not perform as well as expected, achieving a MAE of 0.61 for depth estimate and 0.51 for the velocity estimate.

References

[1] Kala, R. (2016). Advanced Driver Assistance Systems. In R. Kala (Ed.), On-Road Intelligent Vehicles (pp. 59–82). Elsevier.

[2] Lucas, B.D. and Kanade, T., 1981, April. An iterative image registration technique with an application to stereo vision.

[3] Baker, S. and Matthews, I., 2004. Lucas-kanade 20 years on: A unifying framework. International journal of computer vision, 56(3), pp.221-255.

[4] Z. Kalal, K. Mikolajczyk, and J. Matas. Forwardbackward error: Automatic detection of tracking failures. In Proc. ICPR, pages 2756–2759. IEEE, 2010. 2, 3

[5] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3061–3070

[6] C. Vogel, K. Schindler, and S. Roth, "Piecewise rigid scene flow," in Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 1377–1384.

[7] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 2758–2766.

[8] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2462–2470.

[9] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8934–8943.

[10] M. D. Kampelmuhler, M. G. Muller, and C. Feichtenhofer, "Camera-based vehicle velocity estimation from monocular video," in Computer Vision Winter Workshop (CVWW), 2018, 2018.

[11] Akolkar, H., Ieng, S.-H., Benosman, R. (2020). Real-time high speed motion prediction using fast aperture-robust event-driven visual flow. Arxiv.Org.

[12] Song, Z., Lu, J., Zhang, T., Li, H. (2020). End-to-end learning for inter-vehicle distance and relative velocity estimation in ADAS with a monocular camera. In arXiv [cs.CV].

[13] D. Xu, E. Ricci, W. Ouyang, X. Wang, and N. Sebe, "Multi-scale continuous crfs as sequential deep networks for monocular depth estimation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5354–5362.

[14] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep Ordinal Regression Network for Monocular Depth Estimation," in IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[15] G. Brazil and X. Liu, "M3d-rpn: Monocular 3d region proposal network for object detection," in Proceedings of the IEEE International Conference on Computer Vision, Seoul, South Korea, 2019.

[16] Patil, V., Van Gansbeke, W., Dai, D., Van Gool, L. (2020). Don't forget the past: Recurrent depth estimation from monocular video. IEEE Robotics and Automation Letters, 5(4), 6813–6820.

[17] McCraith, R., Neumann, L., Vedaldi, A. (2021). Real time monocular vehicle velocity estimation using synthetic data. In arXiv [cs.CV].

[18] Huang, K.-C., Huang, Y.-K., Hsu, W. H. (2021). Multi-stream attention learning for monocular vehicle velocity and inter-vehicle distance estimation. In arXiv [cs.CV].

[19] Dai, Q., Patii, V., Hecker, S., Dai, D., Van Gool, L., Schindler, K. (2020). Self-supervised object motion and depth estimation from video. 2020 IEEE/CVF (CVPRW)

[20] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox and A. Geiger, "Sparsity Invariant CNNs," 2017 International Conference on 3D Vision (3DV), 2017, pp. 11-20

[21] Zhang, Yunpeng, Jiwen Lu, and Jie Zhou. "Objects are different: Flexible monocular 3d object detection." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

[22] Liu, Z., Zhou, D., Lu, F., Fang, J. and Zhang, L., 2021. Autoshape: Real-time shape-aware monocular 3d object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 15641-15650).

[23] Yu, F., Wang, D., Shelhamer, E. and Darrell, T., 2018. Deep layer aggregation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2403-2412).

[24] Loshchilov, I. and Hutter, F., 2019. Decoupled weight decay regularization (2017). arXiv preprint arXiv:1711.05101.

Appendix

8.1 Monovelocity results:

The monovelocity was re-implemented with reference to their github repo. The results are evaluated on TuSimple dataset and KITTI(depth) dataset are shown in the following tables.

Table 2: MSE for Inter-Vehicular Distance and Inter-Vehicular Velocity

TuSimple				
Type	Our Results	Baseline	Our Results	Baseline
Near	7.86	7.82	0.28	0.12
Medium	6.31	6.10	0.36	0.34
Far	22.12	21.02	0.94	2.09
All	10.24	10.23	0.50	0.86

Table 3: MSE (m^2) for Inter-Vehicular Distance

Kitti		
Type	Our Results	Paper Results
Near	0.28	0.29
Medium	1.01	0.93
Far	1.48	1.57
All	0.96	0.94

8.2 Modified Monoflex results:

The modified Monoflex has different kinds of losses which are plotted in the Appendix. Additionally MAEs are plotted for every depth estimate and compared.

1. **Total Loss:** Total loss consists of all losses including IoU loss for the predicted box, depth loss from direct depth regressor, depth loss from key points, weighted average depth loss, L1 corner loss.
2. **Depth Loss:** Loss calculated for the direct depth regressor.
3. **Key-point Depth Loss:** Loss calculated from the depth estimates calculated by 8 key-points.
4. **Key-point Depth Loss Additional:** Loss calculated from the depth estimates calculated by additional key-points.
5. **Weighted Average Depth Loss:** Weighted average depth loss calculated based on uncertainties in estimates of each kind of loss.
6. **Depth Estimate MAEs:** MAE values for the respective depth estimates.
7. **Key-point MAE:** MAE values for the key-points.

8.2.1 8 Key-points (Approach 1)

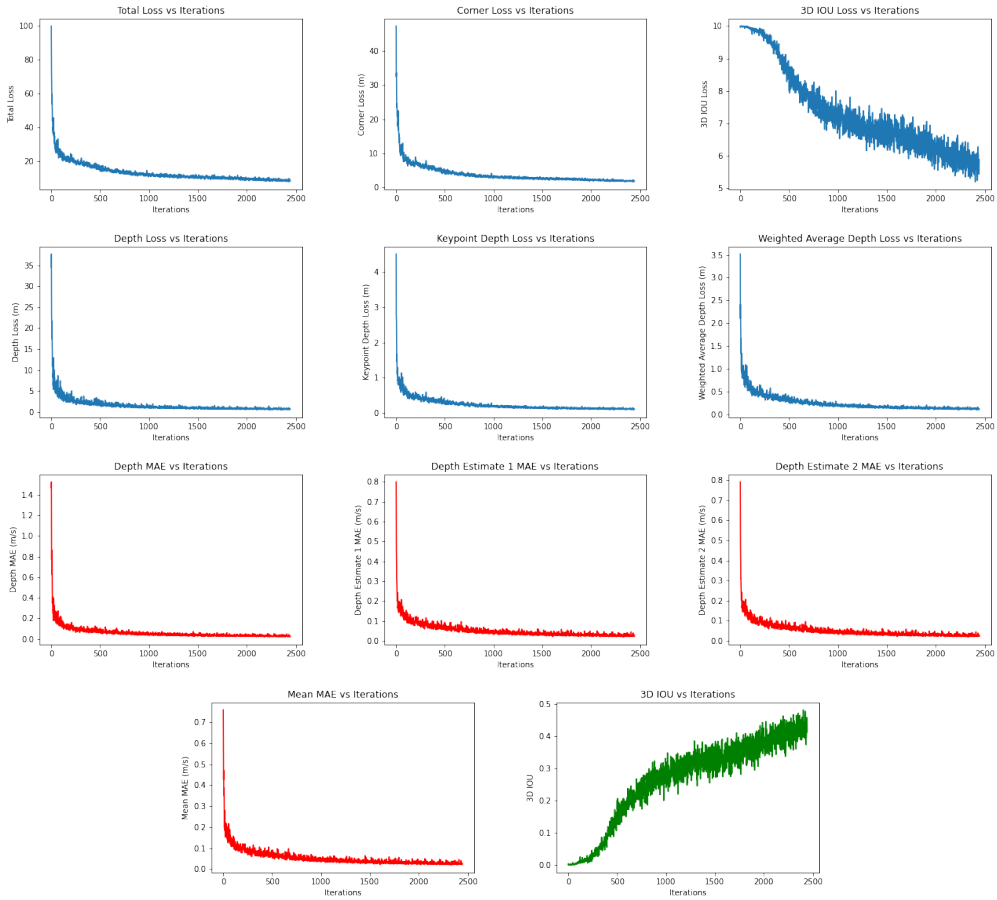


Figure 9: Graphs for Approach 1: The Blue graph show the Loss curves, the Red graphs show the MAE values and the Green graph shows the 3D box IoU.

8.2.2 22 Key-points (Approach 2)

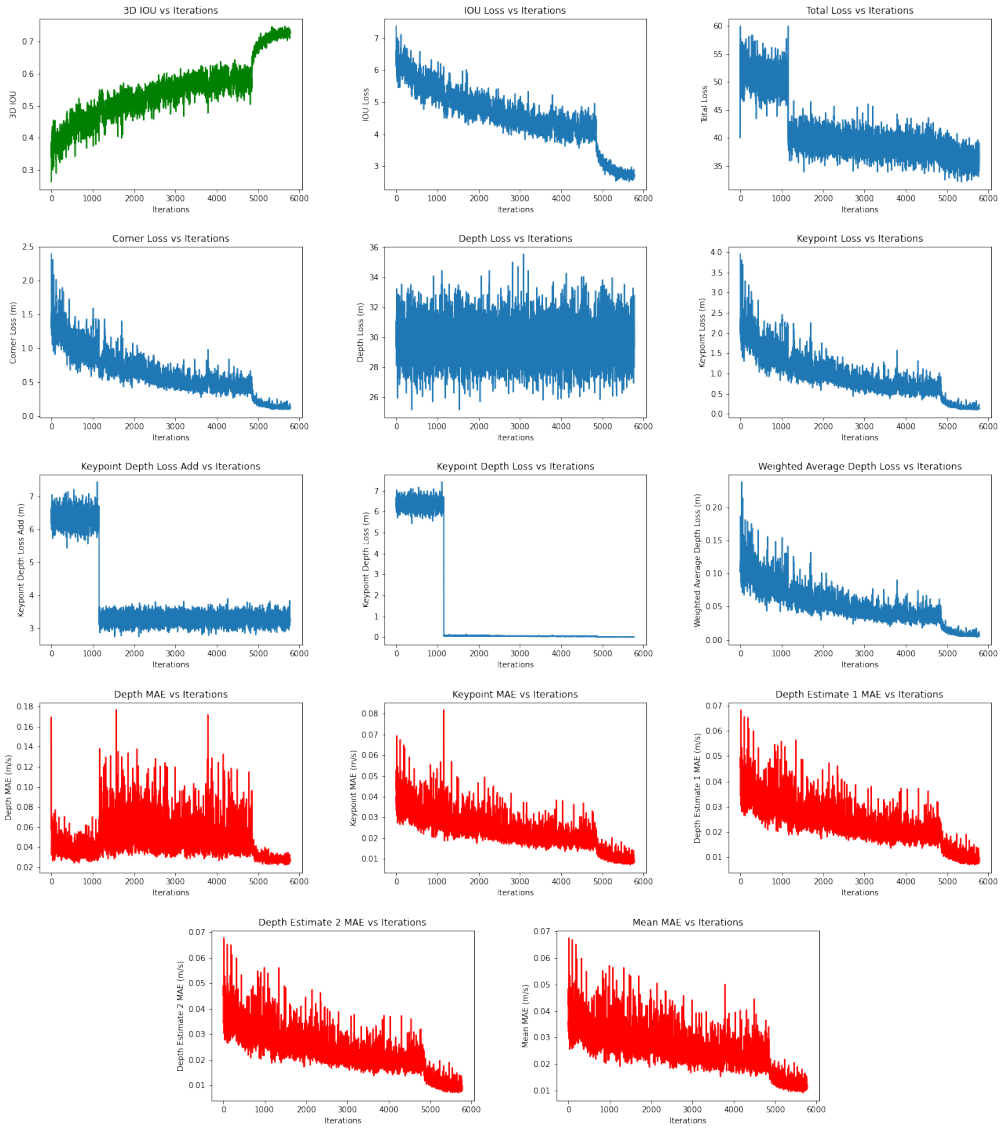


Figure 10: Graphs for Approach 2: The Blue graph show the Loss curves, the Red graphs show the MAE values and the Green graph shows the 3D box IoU.

8.2.3 89 Key-points (Approach 3)

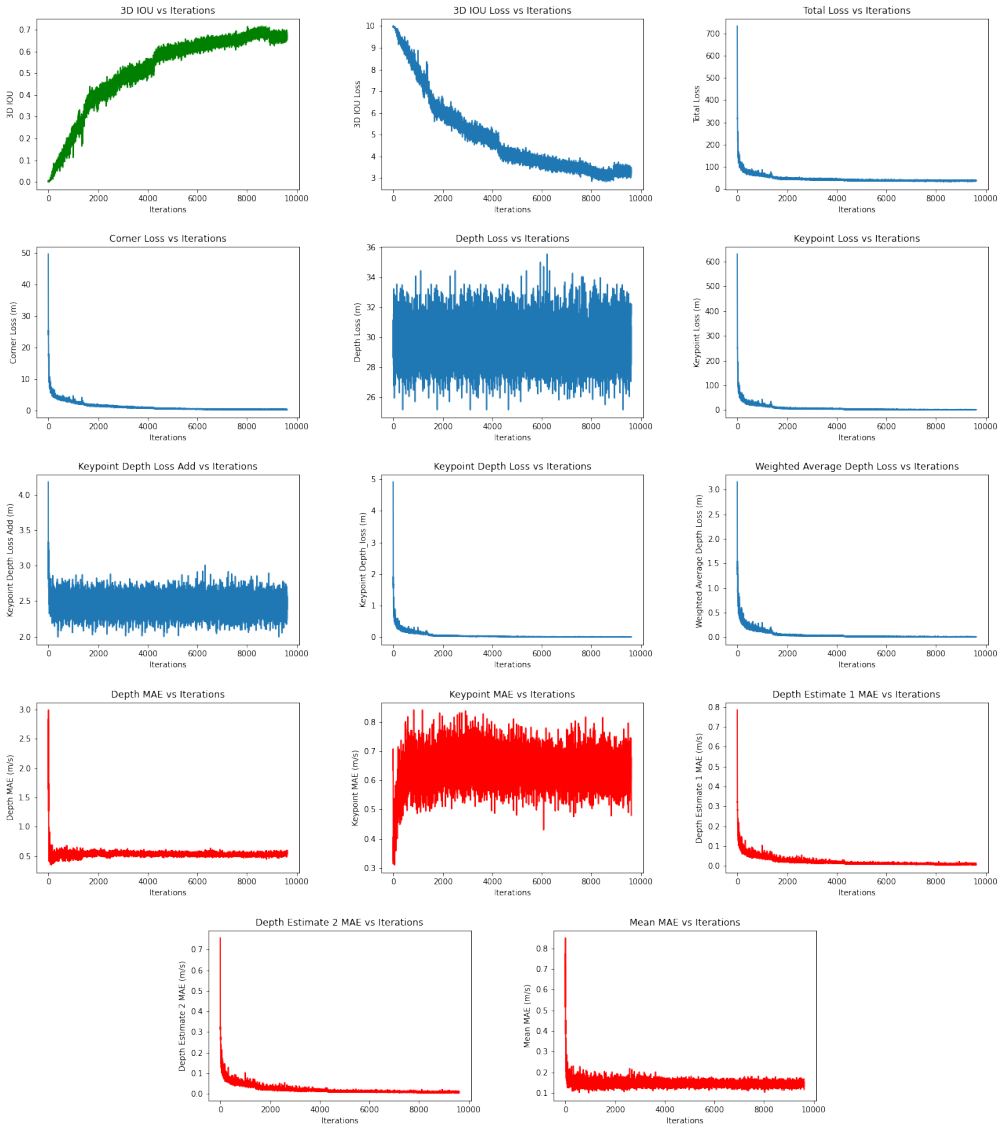


Figure 11: Graphs for Approach 3: The Blue graph show the Loss curves, the Red graphs show the MAE values and the Green graph shows the 3D box IoU.

8.2.4 MViT training graphs

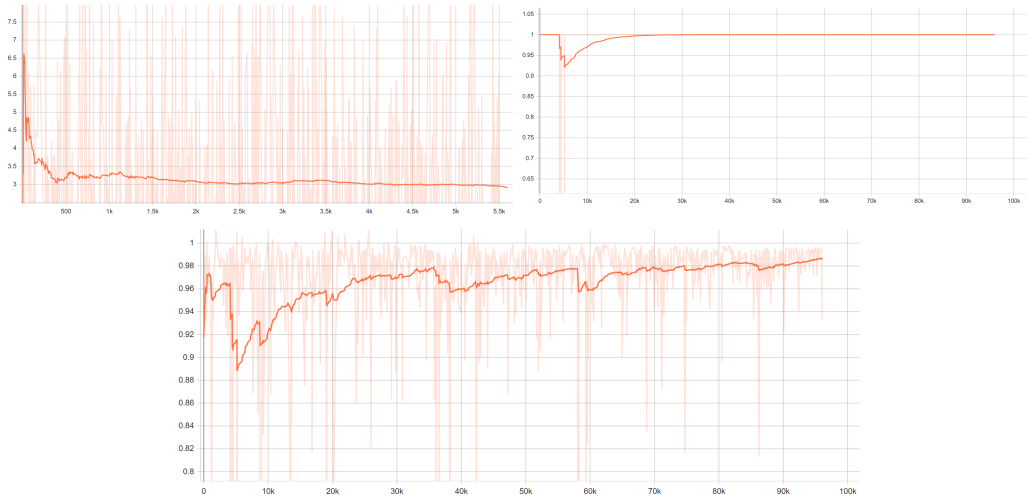


Figure 12: (a) Training loss considering input every two frames. (b) Validation depth MAE considering input every two frames. (c) Validation velocity MAE considering input every two frames.

Figure 13: (a) Training loss considering input every four frames. (b) Validation depth MAE considering input every four frames. (c) Validation velocity MAE considering input every four frames.

8.2.5 Additional details:

A website is created for this project which summarizes the paper. This can be viewed in website.