

# Reduced Supervision for 3D Perception in Autonomous Driving



Robert McCraith  
Kellogg College  
University of Oxford

A thesis submitted for the degree of  
*Doctor of Philosophy*

Trinity 2022

Robert McCraith  
Kellogg College

Doctor of Philosophy  
Trinity Term, 2022

# Reduced Supervision for 3D Perception in Autonomous Driving

Deep Learning methods typically require large amount of labelled data to generalise to real world examples. Autonomous driving is no exception to this with a very long tail of scenarios requiring a safe, dependable prediction. Datasets available for this application are typically captured in a small set of geographical regions which is insufficient for learning useful predictors which would generalise well in other regions where vehicle types, building styles and natural scenery differ substantially. In this thesis I explore alternative strategies for training network which understand autonomous driving scenes without the need to spend substantial resources on human annotation. This will allow data collection from a wider range of environments while retaining highly accurate understanding of the scene.

Firstly I explore synthetic data with properties matching that of real data can be used to train a network to understand allo-vehicle relative velocity. Generating synthetic data means I can simulate cars which move in ways out of distribution compared to those seen in a finite training set.

Next I delve into whether the use of temporally and/or spatially adjacent image captures can provide sufficient information to understand relative depth of objects in a scene. I show as the information required to train is also available at test time I can carefully select layers of the network to fine tune efficiently without loss of generalisation. While this kind of network is used to great success to train relative depth estimation their use in practice is limited owing to their lack of scale which makes them infeasible for use in downstream tasks. With this in mind I show that using only the known height of the camera I can achieve outputs with correct scale without the typical need for additional sensors.

Finally I combine temporal and cross modal information to derive labels for objects in 3D. I show that by taking 2D object detections I can train a 3D LiDAR detector without the need for any of the objects 3D parameters and that even with noisy 2D predictions, heavily occluded or truncated in image space, I can derive accurate detections by ensuring 3D predictions are temporally consistent. Our first approach to this problem utilises instance segmentation predictions at training and test time whereas our second solution demonstrates how the ability to detect can be transferred from the image sensor to LiDAR sensor.



This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfillment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Robert McCraith  
Kellogg College  
October 2022

Copyright © 2022  
Robert McCraith  
All rights reserved.

## Acknowledgements

I am extremely grateful to my supervisors, Professor Andrea Vedaldi and Dr. Lukas Neumann, for many years of guidance. Their support and guidance were integral to my development during my DPhil, and I am truly thankful to have had the opportunity to learn a great deal from them. I would also like to thank Eldar Insafutdinov, who's mentor-ship and insights assisted greatly in the latter years of my DPhil.

The members of the VGG over the last couple of years, of whom there are too many to name also contributed greatly, both academically and socially to my time in Oxford. I have also had the pleasure of making many other friends in my time here from college, rowing, AIMS, and elsewhere all of who I am grateful to have spent time with.

Finally I am eternally grateful to my family for their constant encouragement in all of my studies and other activities throughout the years which made it possible to achieve so many things.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Tasks . . . . .	4
1.2.1	Relative Velocity Estimation . . . . .	4
1.2.2	Depth Estimation . . . . .	5
1.2.3	3D Object Detection . . . . .	6
1.3	Contribution Outlines . . . . .	7
1.3.1	Synthetic Data Generation for Relative Velocity Estimation . . . . .	7
1.3.2	Diving into the details of Monocular Depth Estimation . . . . .	7
1.3.3	Cross-Modal Knowledge Distillation . . . . .	9
1.4	Publications . . . . .	11
<b>2</b>	<b>Literature Review</b>	<b>12</b>
2.1	Relative Velocity Estimation . . . . .	12
2.1.1	Velocity Estimation . . . . .	12
2.1.2	Synthetic Training Data . . . . .	13
2.2	Depth Estimation . . . . .	14
2.2.1	Depth Sensors . . . . .	14
2.2.2	Supervised Depth . . . . .	15
2.2.3	Self-supervised Depth . . . . .	15
2.3	3D Object Detection . . . . .	17
2.3.1	Supervised . . . . .	17
2.3.2	Weakly Supervised . . . . .	19
<b>I</b>	<b>Relative Velocity Estimation</b>	<b>21</b>
<b>3</b>	<b>Relative Velocity Estimation</b>	<b>22</b>

<b>II</b>	<b>Monocular Depth Estimation</b>	<b>30</b>
4	Monocular Depth Estimation with self-supervised instance adaption	31
5	Calibrating self-supervised monocular depth estimation	50
<b>III</b>	<b>3D Object Detection</b>	<b>62</b>
6	Lifting 2D Object Locations to 3D by Discounting LiDAR Outliers across Objects and Views	63
7	Direct LiDAR-based object detector training from automated 2D detections	72
8	Conclusion	82
8.1	Impact . . . . .	82
8.1.1	Relative Velocity Estimation . . . . .	82
8.1.2	Monocular Depth Estimation with self-supervised instance adaption . . . . .	82
8.1.3	Calibrating self-supervised monocular depth estimation . . . .	83
8.1.4	Lifting 2D Object Locations to 3D by Discounting LiDAR Outliers across Objects and Views . . . . .	83
8.1.5	Direct LiDAR-based object detector training from automated 2D detections . . . . .	84
8.2	Future Work . . . . .	84
8.2.1	Relative Velocity Estimation . . . . .	84
8.2.2	Monocular Depth Estimation . . . . .	84
8.2.3	Weakly Supervised 3D Object Detection . . . . .	85
8.3	Conclusion . . . . .	86
<b>A</b>	<b>Statements of Authorship</b>	<b>87</b>
	<b>Bibliography</b>	<b>93</b>

# Chapter 1

## Introduction

### 1.1 Motivation

Deep Learning methods present a technique to transform high dimensional data to compact vector representations which allow us to process data at a much grander scale than traditional sensor processing methods. This has been demonstrated across multiple sensing modalities such as images, audio, and text where sensors can capture large quantities of data quickly in an unstructured manner. For images the use of Convolutional Neural Networks has radically transformed the range of tasks which can now be performed with a high level of accuracy. These methods break down the problem by applying multiple levels of processing with early layers attempting to describe low level features at the scale of a couple of pixels wide while the later layers operate on the information derived by the earlier layers. This approach allows each subsequent step to operate on data progressively more abstract than the previous, with the learning process modifying weights such that each filter develops the ability to detect different attributes in the input which are useful for the desired output. This learning process however requires an increasing amount of data for tasks which continue to get more and more expensive for humans to annotate. Unlike previous techniques deep neural networks require a large amount of training data to generalise well to unseen examples and not overfit the training set. As such datasets constructed for deep learning methods are typically quite large in scale, as these techniques can reason about data at a much more complex level than manually engineered methods. These large scale datasets however require a manual annotation which we train the network to predict for us. In ImageNet[11] and other classification tasks, only one label is needed for each image, however more compelling tasks such as object detection MS COCO[29] requires 5 values per object ( $x, y, w, h$ ) location and size as well as a class label, with many images containing dozens of objects, resulting in much greater

annotation time. Other tasks are also annotated on these datasets such as human pose, image masks, and image keypoints, again these annotations being costly in both time and money. Data collection itself however can be relatively simple, for autonomous driving for example the only requirements would be an array of sensors (typically a combination of RGB cameras, LiDAR and RADAR) with known relative transformations yet all existing datasets in this space have only been captured in small geographical regions and are therefore biased towards vehicles typical of these regions (manufacturer, type, size, colour) which will undoubtedly result in models that generalise poorly to scenes in different locations. These datasets also only contain relatively small numbers of objects when compared to other deep learning tasks. With these problems in mind the ability to train a deep learning model without the need for human annotation will allow autonomous driving solutions which can continually improve their networks understanding as they drive, or even expand an initial training set with examples unlike those previously seen (an important improvement given the long tail of scenarios possible in driving). Datasets such as BDD100K[41] show that dataset collection can be completed in a federated manor, this combined with the ability to automatically label new captures will allow models trained on the most diverse imaginable dataset resulting in networks capable of understanding the more challenging scenes unlikely to be seen in a dataset captured by the small number of capture vehicles driven for a limited number of hours yet might occur reasonably often in some areas.

In this thesis I develop methods for training deep networks to operate on unlabeled data which would allow our methods to operate on a much larger quantity of data without the need to manually annotate the information we wish to reason about. To tackle this difficult problem I explore (1) *synthetic* data, (2) *temporal/multi-view* data, and (3) *Cross-modal* data where information from one sensor is used to assist understanding another sensors data. The applications considered in this thesis include: relative velocity estimation, monocular depth prediction and 3D Object Detection, each representing an important part of reasoning about our surroundings in 3D. The overarching goal is to develop solutions to these problems without the need for a human annotator in the loop, allowing larger and more diverse datasets to be used which will improve the ability of these solutions to generalise to unseen environments.

*Synthetic Data:* Owing to practical limitations on capturing data in the real world many scenarios often observed during driving are not represented. Autonomous Driving however depends on our vehicle protecting its own passenger and those in the

environment which means that understanding our surroundings in the average case is not enough as dangerous events happen rarely in the real world. Deep Learning methods are fantastic at understanding examples in testing which are close to those seen in training, however like many other machine learning methods test cases far from the training set can result in predictions difficult to determine beforehand, a big problem for safety critical applications. With this in mind I use synthetic data in an abstract space to represent data with similar properties to that seen in real world examples while also being extendable to have properties rarely seen in real life but important to develop an understanding of to ensure a dependable prediction is made in these events.

*Temporal/multi-view*: many existing methods rely solely on a single sensor capture and their datasets only train and annotate this information. However in real world usage sensors stream data continuously across time. The additional data across time can be leveraged to provide additional information for our initial frame, or to extend the amount of data which we can learn from. In this thesis I use temporally adjacent frames from a moving ego vehicle to evaluate the accuracy of monocular depth estimates by warping the initial frame to a new viewpoint and comparing the warped image to the actual image capture at this time/location. I also leverage consistency across time and views to develop a richer understanding of 3D object location. This is especially useful when using sparse ranging measurements which at some locations and times can provide inconclusive data only to have ambiguities cleared up in subsequent frames.

*Cross-modal*: some sensing modalities are better studied than others, and each has it's own strengths and weaknesses compared to the others. Image tasks for example have many more datasets with a wider range of images annotated than LiDAR/ranging sensor based tasks owing to the much greater complexity of labeling data in 3D. With this in mind I explore the possibilities of transforming object information across sensing modalities, namely from image space where methods are robust and trained on diverse data to LiDAR where datasets are typically limited geographically and in quantity. I feel that this strategy will allow future work to utilise much larger more diversely collected datasets rather than overfitting to specific geographies represented in current datasets.

## 1.2 Tasks

The following section describes the tasks tackled in this thesis, all of which involve the use of 2D image information to derive information about the 3D world across time.

### 1.2.1 Relative Velocity Estimation



Figure 1.1: TUSimple Relative Velocity Estimation dataset[1] example

When navigating an environment with other objects in motion understanding their motion is a fundamental requirement of planning our future movements. In Figure 1.1 we observe three other vehicles and annotations of their relative position and location. With this information we can estimate the location of these vehicles in the immediate future. Sensors like RADAR are generally used for such tasks as they can easily determine velocity of other objects, however the use of cameras for this task is highly desirable either as a primary sensor, redundancy or to ensure the RADAR sensor is not damaged and outputting bad data. As velocity broken down is just distance and time having an accurate concept of an objects distance would give us the most beneficial derivative of the input sensor data for this task.



### 1.2.2 Depth Estimation



Figure 1.2: RGB image and a predicted depth map

While capturing an image we take rays of light cast through a lens and hitting a sensor. This produces an image where each pixel represents the number of photons captured from illuminated objects in the direction of the ray (potentially with some transparent intermediate objects modifying the colour along the way). This process converts the colours present in the pixels frustum to a single point in the image. When capturing the photons we take a 3D point (or region), cast it's light through the lens and count how many reach the sensor, which tells us what colour that region of the image should be. Given an image with known intrinsic properties it is not however possible to reverse this operation as the pixel coordinate and intrinsic parameters only describe the angle at which the photons reached the lens and not the distance of the object from the camera. For systems that interact with 3D objects this conversion makes the information 2D image based methods produce far less useful as while we may be able to determine the direction of an object relative to the ego vehicle we cannot determine it's distance. With this in mind much work has been done in using convolutional neural networks to predict a pixel-wise estimate of depth as shown in Figure 1.2. This allows us to utilise cameras in place of LiDAR or other depth sensors for a lower cost system, or provide a fallback option incase of sensor failure due to mechanical problems or adverse weather which may effect different sensors much worse than RGB cameras. Having accurate knowledge of the distance to an object would allow better velocity estimation and object detection in 3D.

### 1.2.3 3D Object Detection

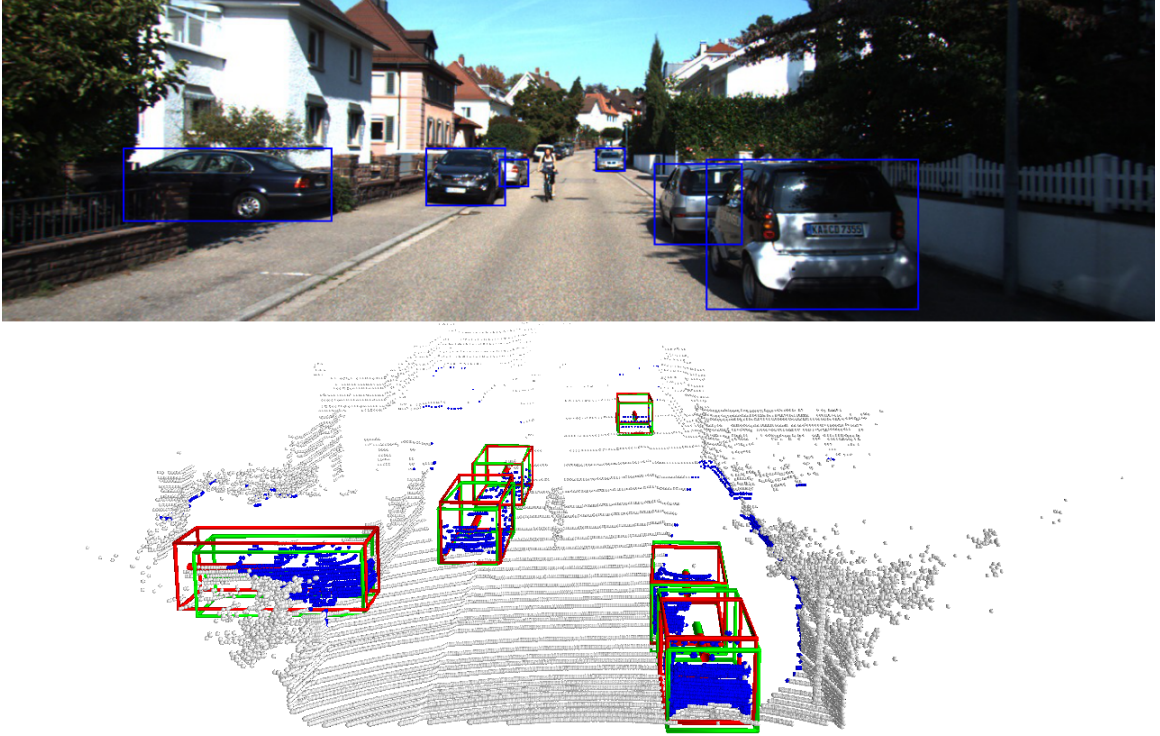


Figure 1.3: 3D Object Detection

Object detection in images is well studied with methods such as Mask R-CNN[23], YOLO[34], and DETR[6] providing reliable 2D object detections for existing large datasets. In autonomous driving however detection of objects in 3D is far more important as it allows us to plan our own future movements and rotation gives a strong prior on which direction we can expect other objects to move. 3D Object Detection (3DOD) however is much more complex to annotate than 2D object detection. Image object bounding boxes have 4 degrees of freedom (top, left, width, height) coordinates. To generate the 3D bounding box annotations and predictions shown in Figure 1.3 we need 7 to 9 parameters (X, Y, Z) center, (Length, Width, Height) size, and (Roll, Pitch Yaw) rotation parameters, often only Yaw is recorded in autonomous driving applications). Human annotation in point clouds is error prone as [15] notes and extremely time consuming (up to 114 seconds per object as noted in [30]). With such a high demand on human annotation time it isn't a surprise that large scale annotated datasets of 3D objects are less common than 2D object detection which ultimately decreases our ability to train robust 3DOD systems.

## 1.3 Contribution Outlines

This section summarises the main contributions of this thesis and outlines the chapters.

### 1.3.1 Synthetic Data Generation for Relative Velocity Estimation

Relative velocity estimation is a fundamental task for autonomous driving where decisions need to be made several seconds in advance in anticipation of the future locations of actors in the environment. This requires analysis of the recent history of objects to determine their relative velocity compared to the ego vehicle which gives a strong prior on future locations. While datasets such as TuSimple[1] allow us to build a model for these tasks it is difficult to truly capture the variance of velocities possible in real driving in a finite dataset yet we must train a model which can make reasonable predictions in all cases. In Chapter 3 we explore the attributes of this dataset and leverage some simple insights to develop a dataset of completely artificial data with realistic properties. The key insight of this work is that a sequence of bounding boxes is sufficient to adequately describe the relative motion of allo-vehicles. In contrast to other synthetic datasets such as [12, 17] where the entire scene must be simulated which greatly limits the amount of data which can be generated, our simple transformation into an abstract space allows us to create a far broader range of examples with properties both within the range seen in the real data and also far exceeding the samples seen during the dataset captured, yet physically plausible.

### 1.3.2 Diving into the details of Monocular Depth Estimation

To understand velocity we necessarily need an understanding of depth. This however does not come naturally to monocular vision systems. With these limitations of cameras in mind we next focus on predicting pixelwise depth. In previous works [20, 43, 21] a sequence of frames from a single camera have been used to train a model which jointly learns depth for a single frame and the relative pose change to another frame in the sequence. This technique has shown some impressive results, approaching those of methods supervised by LiDAR sensors (by interpolating the sparse depth maps when LiDAR points are projected into an image captured at the same time). While deep learning methods utilise strategies such as augmentation, shuffled mini-batches, Dropout and Batch Norm the parameters learned at the end of training still perform better on training set examples compared to test samples.

Given that in real world deployments of such methods the temporally adjacent frames are also present this raises the question of whether we can improve the performance of our model at test time while not changing the parameters too much and getting outputs which have overfit the specific frame as the outputs may have degenerated owing to the prerequisites of such methods being violated. Indeed papers such as GLNet[8] have shown that such an idea is possible on an individual test time frame, their method however is far too slow for a real world use, with a single images fine tuning taking 2 seconds or 40 seconds depending on the exact strategy used. In Chapter 4 we explore which network components influence poor generalisation at test time and how such parameters can be fine tuned to improve performance while not overfitting the specific example or taking too long to tune. In this paper we show even using less time than the quickest strategy used in [8] we still achieve substantially better performance, and that with fewer update steps we still improve the baseline substantially. We further demonstrate that while until now these fine tuning steps have been deployed on a single test time frame with the parameters then reset to the pre-trained values the best way to utilise this technique is actually to leverage the similarity of frames in a long sequence and slowly tune parameters. This means that rather discarding the scene specific knowledge we gained in the previous frames fine tuning we can perform fewer steps while still improving performance substantially, making these techniques usable in real time applications.

Despite the excellent performance numbers quoted by existing monocular depth estimation networks the information learned only tells us the relative depth of one object in the image compared to another. This manifests in the evaluation where the frame by frame LiDAR data is needed to correct the scale of the networks output values. This severely limits the utility of such networks for downstream tasks. In PackNet[22] the ego velocity is used to calibrate the magnitude of the pose networks predicted translation vector. This results in the depth networks scale to be calibrated properly as during the calculation of the image warping the predicted depth map is converted into a point cloud, transformed by the estimated relative pose and projected into the camera at the new viewpoint. Having the magnitude of the translation correct forces the depth values to use the correct scale. We however find in Chapter 5 that the known camera height can also be used to calibrate the depth maps values to the correct scale. The advantages of this that we don't require IMU data to train the model, this can be beneficial as wheel odometry is unreliable and GPS based systems struggle to get a consistent signal in many areas, and even when they do the data is often too noisy to use for supervising a network requiring highly precise values. This

work was completed in parallel with [36] who use a similar approach. In Chapter 4 and Chapter 5 we use a state of the art architecture [21] and loss functions and focus on the optimization strategy for to enable quick learning at test time and add a scaling method which can be utilised without additional test time sensors.

### 1.3.3 Cross-Modal Knowledge Distillation

A good understanding of depth gives us a much greater ability to determine the relative position of other objects detected in our image. Even with the past few years of development monocular self-supervised and supervised depth estimation still struggles to assist vision only estimation of 3D bounding boxes as seen in [27]. With these limitations in mind we pivot to utilising LiDAR as a source of accurate depth information. However compared to images LiDAR is much more challenging to annotate. Images have been annotated for classification, detection, panoptic segmentation and inter frame tracking to name but a few tasks. These annotations require no additional sensor data and therefore datasets can be constructed with images from a wide variety of sources. LiDAR on the other hand is far less prevalent and much more difficult to annotate accurately owing to partial object scans, sparsity at increasing distances and ambiguous point clouds (see [15] for discussion of LiDAR annotation difficulties). While some datasets capture both RGB and depth data (NYU[31], ScanNet[10], Matterport3D[7]) in those cases the depth information is only used to evaluate depth estimation from the RGB image. Even recent datasets captured for autonomous driving specifically focus more on image based tasks. BDD100K [41], Cityscapes[9] and NuImages[4] all capture a large number of frames from a variety of regions and annotate for object detection, panoptic segmentation, lane markings, and drivable area yet none of these provide any 3D information, arguably far more important space for scene understanding in autonomous driving. Even more confusingly we find that datasets which use LiDAR and annotate in 3D such as NuScenes[4], Waymo[35] and KITTI[19] contain no 2D annotation for 3D objects (NuScenes) or annotate only a 2D box for each 3D annotated object without any semantic segmentation or tracking across cameras (Waymo and KITTI).

Even the biggest autonomous driving datasets with annotations in 3D contain a relatively small and non-diverse number of objects compared to image datasets. Furthermore in image based tasks sensors are typically similar enough that a network which works on one cameras output will also operate with a similar level of accuracy to the camera it was captured on (as data is often crowd sourced this attribute essentially comes for free). Autonomous driving platforms however often use sensor

configurations which vary greatly from one manufacturer to another, LiDAR alone can come in a wide variety of settings with sometime significant variance in the frequency of sweeps, number of beams (vertical resolution) and accuracy rating at a given distance meaning that deploying a model trained on one platform to another might not be as effective for LiDAR data as it is for RGB data. Most proposed hardware configurations for autonomous driving suggest a combination of RGB cameras and LiDAR/RADAR sensors. As models trained on images transfer across different RGB cameras but LiDAR does not it would be advantageous to utilise the image based predictions to reason about the data captured by the LiDAR sensor. This has multiple benefits, namely that image based problems are well studied with many mature methods with highly accurate predictions on a wide variety of data and secondly that if the image can assist the LiDAR then the specifics of the LiDAR sensor are no longer as significant as we can re-train our network with the new sensor configuration and potentially adopt even perform some fine tuning as we explored in monocular depth. This strategy could also be used to greatly accelerate the creation of new datasets on novel platforms, or to continually expand an existing dataset to cover new target objects, geographies or improve performance on existing objects with more training data. The concept of using the RGB image detections to reduce the search space in point clouds is explored in Chapter 6 where I perform detection in the image space then determine the precise location by training a network using the LiDAR points which project into the instance mask. A similar pre-processing is used in Chapter 7 where instead of detecting in the image we only use the image instance segmentation obtained to construct the loss, meaning our end product can operate only on LiDAR data at test time.

## 1.4 Publications

The work presented in this thesis is as follows:

- Chapter 3 Relative Velocity Estimation
  - R. McCraith, L. Neumann, A.Vedaldi
  - IEEE Intelligent Vehicles Symposium 2021
- Chapter 4 Monocular Depth Estimation with self-supervised instance adaption
  - R. McCraith, L. Neumann, A. Zisserman, A.Vedaldi
- Chapter 5 Calibrating self-supervised monocular depth estimation
  - R. McCraith, L. Neumann, A.Vedaldi
  - Machine Learning for Autonomous Driving, NeurIPS 2020 Workshop
- Chapter 6 Lifting 2D Object Locations to 3D by Discounting LiDAR Outliers across Objects and Views
  - R. McCraith, E. Insafutdinov, L. Neumann, A.Vedaldi
  - International Conference on Robotics and Automation (ICRA) 2022
- Chapter 7 Direct LiDAR-based object detector training from automated 2D detections
  - R. McCraith, E. Insafutdinov, L. Neumann, A.Vedaldi
  - In review: Machine Learning for Autonomous Driving, NeurIPS 2022 Workshop

A note on presentation. This thesis is presented as an integrated thesis where the publications are reproduced in the format they were submitted for publication at the respective venues.

# Chapter 2

## Literature Review

### 2.1 Relative Velocity Estimation

#### 2.1.1 Velocity Estimation

Previous works on ego and allo velocity estimation have followed the prevailing trends of computer vision at the time. Early methods for vehicle object detection from videos such as [39] relied on SIFT features to estimate ego velocity, performing 3D scene reconstruction and deriving object motion in images by determining which 3D points violate epipolar constraints for static objects and grouping collections of such points with similar motion to create an object bounding box. More recent methods such as [14] use a Convolutional Neural Network (CNN) to gather object detections and compares similarly compares keypoints from one image to the next relying on violation or epipolar constraints to determine independent motion.

Both of these methods only determine whether the object is in motion and not its actual velocity, for this we can look to [26], as shown in Figure 2.1 they pass a sequence of images through the Median Flow tracker[25] to track the same vehicle in each frame. This bounding box sequence is then used to crop monocular depth estimation, and optical flow estimates over the entire image which are then feed into a fully connected neural network which predicts the velocity. This approach utilises existing monocular depth estimation [20] and optical flow [24] networks pretrained on larger datasets, freeze their weights and only train the final small Multi Layer Perceptron (MLP) on the much smaller TuSimple Dataset[1], allowing them to benefit from these additional inputs while still not overfitting on a dataset relatively small for deep learning methods.



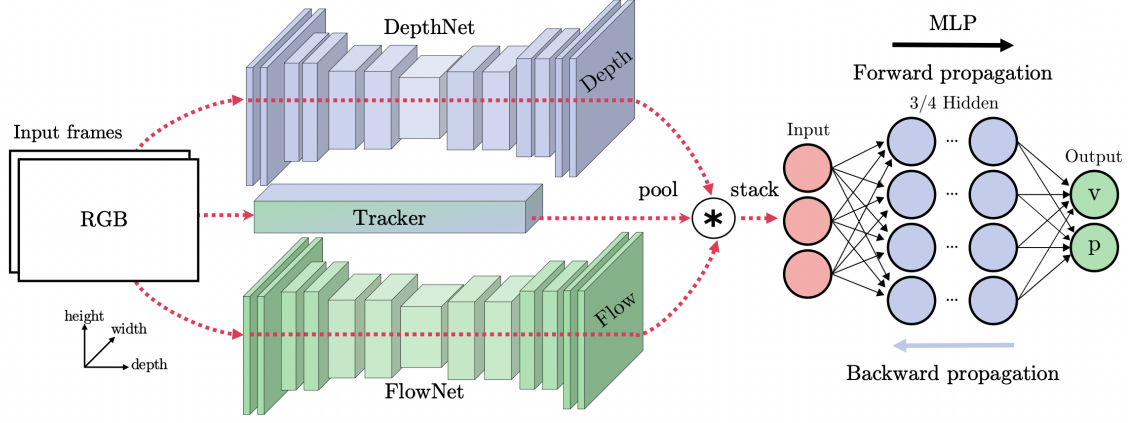


Figure 2.1: Method of [26]

### 2.1.2 Synthetic Training Data

Synthetic training data for autonomous driving has also seen adoption in the age of deep learning methods as it allows the construction of datasets which more precisely capture ground truth and in some cases additional information which is not easy to capture in the wild. In autonomous driving the most commonly used examples of such datasets include Virtual KITTI [18] which attempts to make a virtual clone of the KITTI[19] dataset which allowing the authors to easily produce additional ground truth data such as optical flow, instance segmentation, or even change the weather to produce more diverse challenging data. CARLA[12] provides a simulated environment where textures of the environment and weather conditions can vary, pedestrians can have precise ground truth skeleton annotations and various traffic scenarios can be simulated and captured with a wide array of sensors. Parallel Domain[2] provides many of the same features as CARLA and focuses on higher fidelity rendering, more precise control, more complex annotations and a longer tail of simulated events.

## 2.2 Depth Estimation

### 2.2.1 Depth Sensors

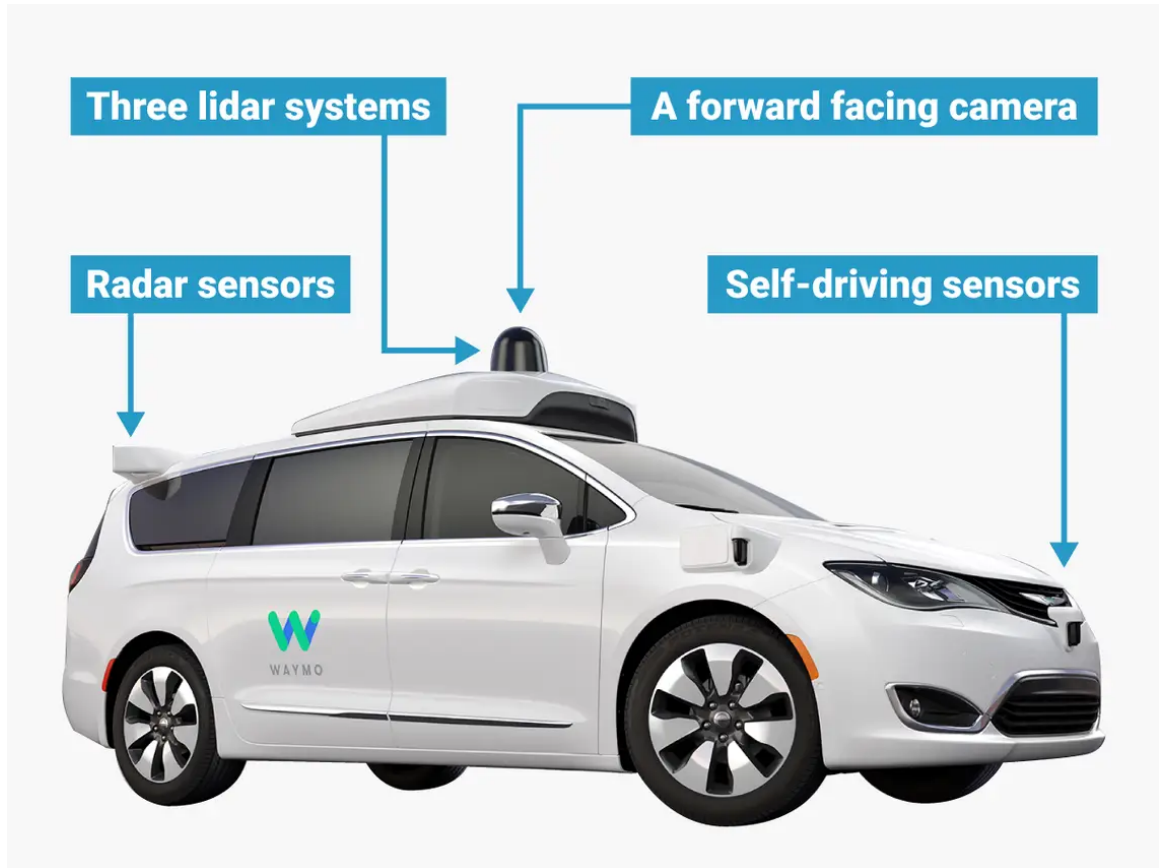


Figure 2.2: A typical autonomous driving platform has multiple sensors often even of the same modality with different viewpoints

When capturing images with a monocular camera the depth information is lost with the only world information retrievable for a given pixel is a ray cast from the camera optical centre. A solution to this could take the form of a calibrated set of multiple cameras, this however relies on the ability to get correspondences between points in each image that represent the same world location and triangulating the point in 3D. This however is difficult in some cases as often areas have low texture or non lambertian surfaces may have a very different appearance in each camera making pixel to pixel matching difficult, thus reducing the number of 3D points retrievable. These challenges have motivated the development of other sensors focused on capturing the 3D location of points in 3D in the environment. In many applications RGB-D cameras have become popular, perhaps most famously in the form of the Xbox Kinect,

the infrared dots projected by these sensors however struggles even in moderately illuminated outdoor scenery. RADAR and LiDAR have emerged as the leading sensors in autonomous driving, with placements of such sensors generally similar to the example in Figure 2.2. Each of these however has its own limitations, RADAR for example has difficulties with reflections and interference from other transmitting sensors, they also struggle with small or slow moving objects. LiDAR produces a representation of a scene which is much sparser for further away measurements, involves more damage prone moving parts, operates at a slower refresh rate than many others (10-15Hz), and can have significant problems with some weather conditions[3] such as fog which causes the point cloud generated to essentially be reduced to noise within a couple of metres of the sensor.

### 2.2.2 Supervised Depth

Most modern techniques for monocular depth estimation rely on transformation of LiDAR point clouds to the viewpoint of a camera, then projecting the points into the image and interpolating the empty pixels using the projection of the LiDAR points. Eigen et al. [13] who predict a coarse depth estimate with deeper features and then refine these measurements by concatenating the coarse predictions to higher level fine predictions which have retained more precise spatial information. Xie et al. [38] used skip connections similar to pass higher resolution but earlier features to layers closer to the high resolution output. DORN [16] removes sub-sampling in the last few layers to retain spatial information, employs a full image encoder and changes the output from the typical regression to ordinal regression in a space-increasing discretization (closer depth values are denser and further away values are sparser).

### 2.2.3 Self-supervised Depth

A alternative to capturing LiDAR alongside RGB images has been to utilise a sequence of images from a moving camera or stereo pair and predict both camera pose differences and depth in a single frame. The depth is then converted into a point cloud which is transformed by the estimated/known camera pose change to the other view. The transformed point cloud is then reprojected into the image at which point we can construct a warping of the pixels from the original image to the other viewpoint. This means the evaluation of the estimated depth and pose can be done in image space. This approach has been popularised by works such as Monodepth [20] which warped the image between a stereo pair of cameras and computed a loss using

pixel intensities, SSIM[37] and a disparity smoothness loss to encourage the disparities to be locally smooth. SfMLearner[43] which removed the need for stereo pairs and performed this process along a video sequence by predicting the pose change between pairs of images rather than depending on the known stereo baseline and uses an explainability map to predict which areas are less reliable as the underlying objects are moving which results in a correctly warped image still being incorrect as some observed objects have moved. Monodepth v2[21] the basis architecture for Chapter 4 and Chapter 5 is depicted in Figure 2.3 refines the loss further by warping forward and backwards and taking only the lower loss which means we only penalise the pixel if the observed object hasn't become occluded, auto masks pixel of objects which haven't moved between the two frames (similar motion to the ego car) and multi-scale appearance loss which up-scales to the original resolution to reduce depth artifacts. In [22] downsampling is replaced by conversion of spatial information to additional feature channels and back again in the decoder steps. This work also supervised the magnitude of the pose translation prediction to match the ego vehicle velocity, thus training depth estimates with realistic scale in the network, resulting in no need for additional depth sensors for scaling at test time.

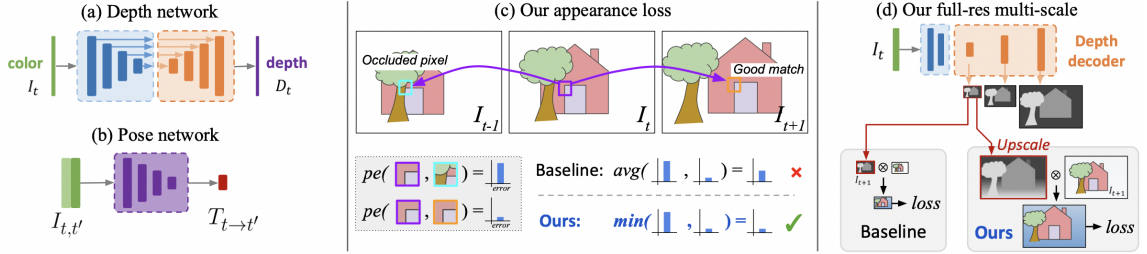


Figure 2.3: Monocular v2[21]

## 2.3 3D Object Detection

### 2.3.1 Supervised

Object detection in 2D images is a well studied problem with many large scale datasets and a wide array of methods. In robotics applications however 3D Object Detection (3DOD) is often a more desirable output as it allows planning of future ego motion and a greater understanding of the spatial relations of objects in a scene. In autonomous driving platforms a LiDAR sensor is commonly used, often to complement a camera of collection of cameras. Frustum PointNet [32] takes a 2D image detection and generates a frustum shown in red in Figure 2.4 and reduces the LiDAR points to only those falling within the frustum. They then perform segmentation on the remaining 3D point cloud to further reduce the outlier LiDAR points to only those within the objects 3D bounding box. Finally they regress the 3D object centre, rotation, and size. We take the frustum pointnet approach to 3D Object Detection in Chapter 6 where detection is performed on the image and location in 3D is predicted by the filtered LiDAR points.

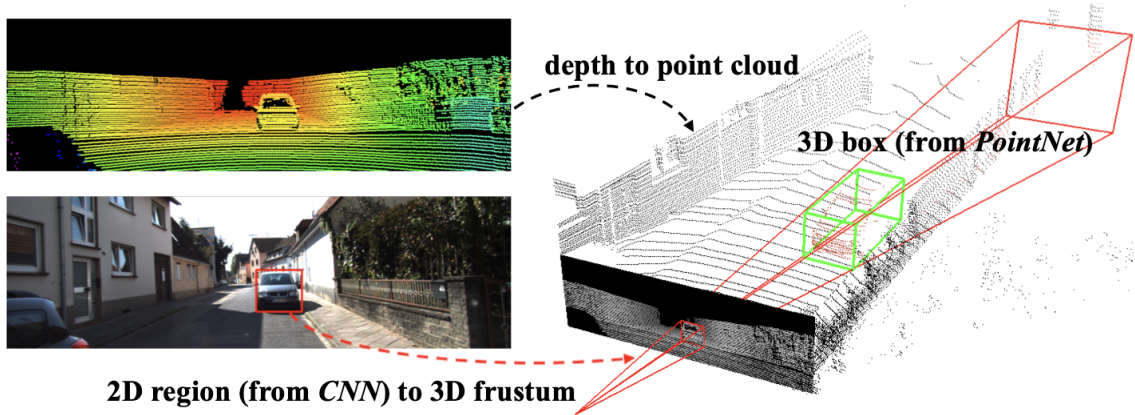


Figure 2.4: Frustum PointNet[32]

Frustum PointNet falls into the image first approach to 3D Object Detection, the other common family of approaches are LiDAR only, a popular direction originating with the architecture of VoxelNet.

VoxelNet[44] takes the LiDAR point cloud and separates its points into voxels. A Feature Learning Network shown in Figure 2.5 then transforms these subsets of points into feature maps, which are scattered into the voxels original location and viewed from above to create a Birds Eye View (BEV) feature map, then a Single Shot Detector is used to perform the object detection. This approach allows the

unstructured point cloud to be converted into a structured representation and when the voxel features are converted into an BEV image regions without LiDAR points can be treated as empty meaning sparse 2D convolutions can be used to save time in the forward pass compared to 3D convolutions. The network outputs a score of each location, a regression to the precise location, size and a rotation offset from the anchors which share the same coordinate in BEV.

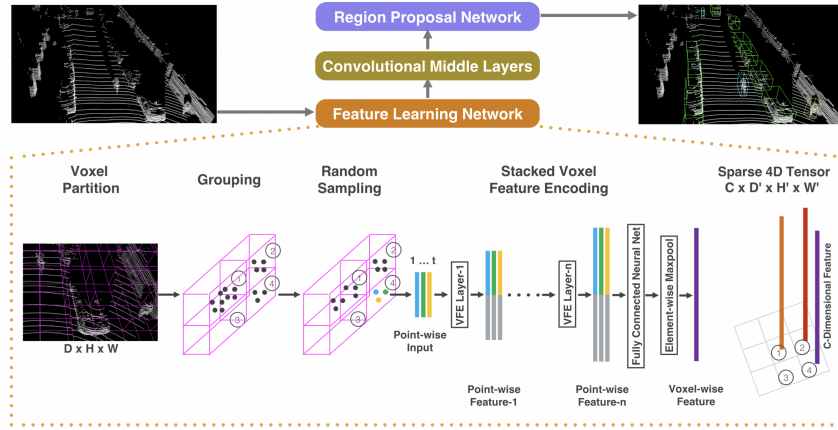


Figure 2.5: VoxelNet [44]

PointPillars[28] changes out the voxels for Pillars (voxels of infinite height) and converting the rotation prediction to a classification task across equally spaced bins.

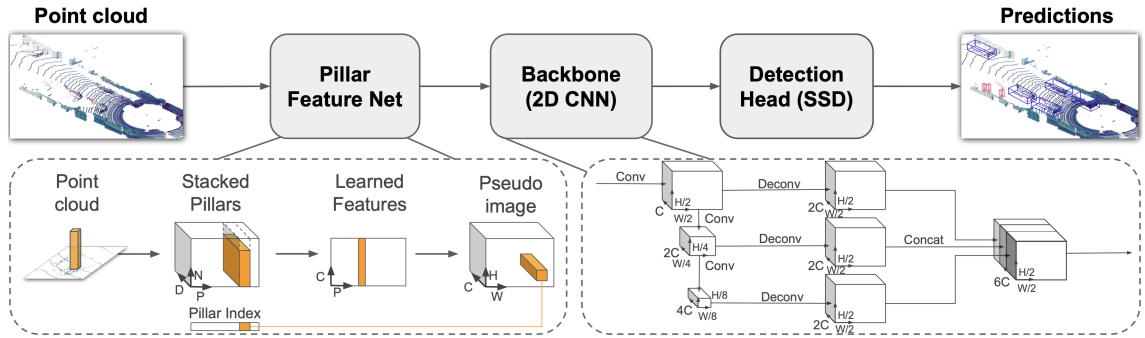


Figure 2.6: PointPillars[28]

CenterPoint[40] removes the orientation associated with predicted location with the output being a heatmap of objectness scores by drawing gaussians on an image and predicting the scores. This technique allows the network to learn a rotation invariant representation of the objects. They also accumulate LiDAR point clouds across time by transforming them to the coordinate frame of the first frame and appending a frame id to the coordinates input to the network which allows the learning of velocity in

newer datasets. This velocity estimation is used for tracking by predicting a sequence of frames and greedily associating objects across predictions.

### 2.3.2 Weakly Supervised

Annotating Data for 3D Object Detection is very challenging compared to many other computer vision tasks. In [30] they attempt to reduce the annotation requirements from the full  $(X, Y, Z)$  centre, yaw and  $(l, w, h)$  size and instead label just the  $X, Z$  centre in birds eye view for most cars, pre-train a network on this data then fine-tune on a smaller set of cars with the additional parameters.

Qin et al. [33] generate 3D anchor boxes which they project into the image along with the LiDAR point cloud. The count of LiDAR points inside the 3D box is then normalised by distance and if this quantity exceeds a threshold the anchor is considered an object proposal. The RGB image is then cropped to the projection of the proposed 3D box. These image crops are then evaluated by a pre-trained teacher network which can confirm the presence of a car and give a rotation estimate during training and teaches the student network to perform these tasks at test time.

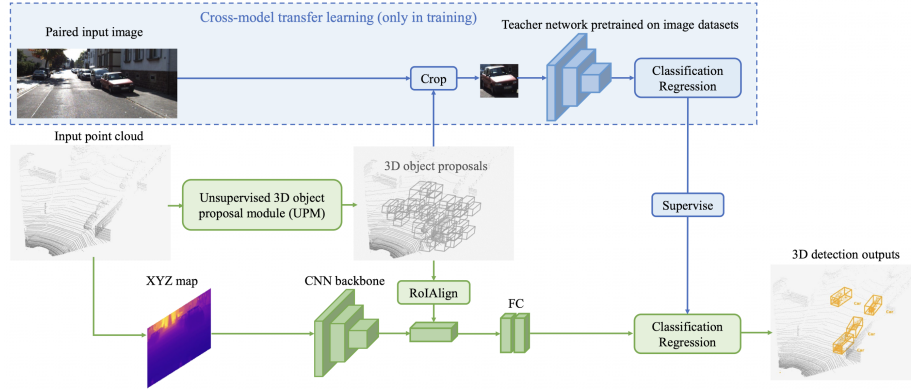


Figure 2.7: Weakly Supervised 3D Object Detection from Point Clouds[33]

Koestler et al.[27] uses instance segmentation and monocular depth estimation to place a deformable car mesh in the candidate location and evaluate its location by comparing to the image and depthmap. By differentially rendering the mesh into the image they can perform small deformations of the mesh to improve its explanation of the instance segmentation it corresponds to without causing a change to the location prediction. Similar to self-supervised monocular depth estimates they also utilise information from adjacent frames to improve results. The depthmaps can come from self-supervised monocular depth predictions but much greater accuracy can be achieved by LiDAR supervised depth estimates.



Zakharov et al. [42] apply a shape differential renderer to a signed distance field representation together with a normalized coordinate space trained initially on a synthetic dataset. After this synthetic initialization real world images are added to the training set getting progressively more challenging (based on the curriculum shown in Figure 2.8) with increasing training steps, allowing the network gradually adapt to real world images. This curriculum allows the knowledge gained in the synthetic pre training step to be retained but also requires knowledge of the difficulty of the 3D objects, requiring some level of manual annotation.

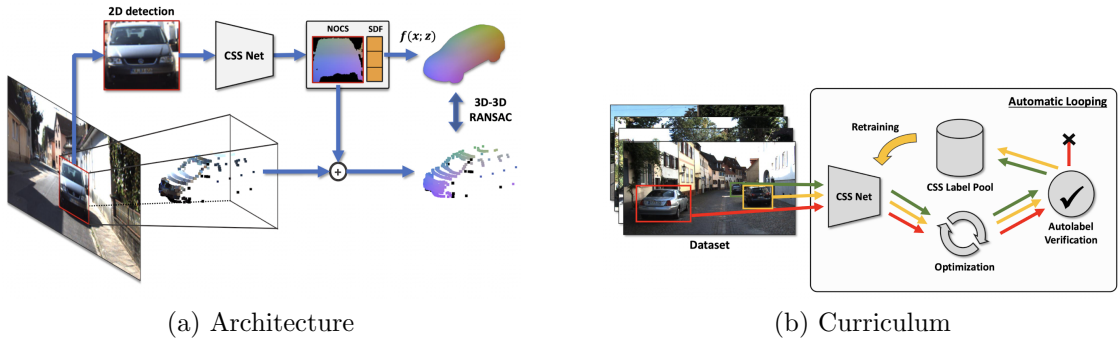


Figure 2.8: Autolabeling 3D Objects With Differentiable Rendering of SDF Shape Priors[42]

While these weakly supervised methods perform admirably given the lack of human annotation in 3D, compared to supervised methods the accuracy of these techniques is significantly worse than supervised methods and as such typically they only report less strict evaluation metrics to obscure the difference to existing work which always reports at a stricter threshold. Some of these methods also depend on some information additional to just a 2D detection, namely [33] teacher network contains yaw predictions, and [42] uses not only a complex synthetic dataset to pre train but also uses KITTI’s object difficulty to first train on easier examples when adapting the network to real images.



# Part I

## Relative Velocity Estimation

## Chapter 3

# Relative Velocity Estimation

This work was presented at IEEE Intelligent Vehicles Symposium 2021 and was a best paper finalist.

# Real Time Monocular Vehicle Velocity Estimation using Synthetic Data

Robert McCraith<sup>1</sup>, Lukas Neumann<sup>1</sup>, Andrea Vedaldi<sup>1</sup>

**Abstract**— Vision is one of the primary sensing modalities in autonomous driving. In this paper we look at the problem of estimating the velocity of road vehicles from a camera mounted on a moving car. Contrary to prior methods that train end-to-end deep networks that estimate the vehicles’ velocity from the video pixels, we propose a two-step approach where first an off-the-shelf tracker is used to extract vehicle bounding boxes and then a small neural network is used to regress the vehicle velocity from the tracked bounding boxes. Surprisingly, we find that this still achieves state-of-the-art estimation performance with the significant benefit of separating perception from dynamics estimation via a clean, interpretable and verifiable interface which allows us distill the statistics which are crucial for velocity estimation. We show that the latter can be used to easily generate synthetic training data in the space of bounding boxes and use this to improve the performance of our method further.

## I. INTRODUCTION

Autonomous driving systems rely on a wide array of sensors including LiDARs, radars and cameras. LiDAR sensors are especially good at estimating the position and velocities of vehicles and obstacles (0.25m/s at the distance of 15 meters [16], 0.71m/s at a wider range of distances [1]). However, LiDAR sensors are also very expensive, not reliable in adverse weather conditions [19] and easily confused by exhaust fumes [9]. Depending on a single source of information in a context such as autonomous driving is also inherently fragile. Hence, it is natural to develop alternative sensors either as redundancies or to improve the accuracy and robustness of sensors like LiDARs. Cameras are a natural choice as they are very cost effective and in principle sufficient for navigation given that humans drive cars primarily using their sense of vision. Furthermore, there is a substantial amount of computer vision research that is directly applicable to autonomous driving.

In this paper, we aim to predict the velocity of other cars based only on a video from a standard monocular camera. Because the vehicle where the camera is mounted on (the *ego-vehicle*) is typically moving as well, the velocity estimate of other vehicles is relative to the velocity of the ego-vehicle.

Our main contribution is to show that, for this problem, one can separate perception from velocity estimation by first mapping the visual data to a mid-level representation — the space of vehicle bounding boxes — with no loss in performance compared to much more complex estimation approaches. The velocity estimation problem is then modelled purely on bounding boxes, which change their position and

size over time, tracking the motion of vehicles on the road in a simplified view of the data. Using such an approach, our simple model outperforms the winning model [15] of CVPR 2017 Vehicle Velocity Estimation Challenge [1], which is a much more complex model that combines tracking with two different deep networks for monocular depth and optical flow estimation.

A major advantage of using such a simple intermediate representation is that it becomes much easier to *simulate* training data for the velocity estimator model. Still, we show that doing so in an effective manner requires to capture accurately the statistics of real bounding boxes. Our second contribution is thus to show how such a synthetic dataset is created, by extracting the necessary data priors from a small set of real data and using it to generate a much larger training set in the mid-level feature space. We then show that training only using this synthetic dataset results in excellent performance on real test data. In the process, we also distill the data statistics that are crucial for velocity estimation from visual data, clarifying in the process what is important and what information is not for this task.

The ability to easily generate synthetic data is not only beneficial to improve velocity estimation accuracy at virtually no cost for existing scenarios, but can be also used to train the model for different driving scenarios, such as different countries, and to model different driving styles and various emergency situations (e.g. unexpected breaking) without the need for laborious real data collection or for expensive 3D photo-realistic animation in order to cater for such situations.

Our approach can also be seen as encapsulating perception, which is often implemented by opaque and difficult-to-diagnose components such as deep convolutional neural networks, in a module which has a simple, interpretable, and testable interface. Decomposing systems in modules that are individually verifiable and that can be connected in a predictable manner is essential for autonomous systems to meet reliability standards such as ISO 26262 [12], [21]. Hence, while end-to-end trainable systems may be conceptually preferable, decompositions such as the one we propose may be essential in practice — fortunately, as we show, this may come with no loss of performance.

The rest of the paper is structured as follows. In section II, prior work is discussed. In section III, we introduce the method, in section IV we discuss the data generation. The evaluation is presented in section V and the paper is concluded in section VI.

Authors are members of the Visual Geometry Group in the University of Oxford, UK. emails: robert, lukas, vedaldi@robots.ox.ac.uk

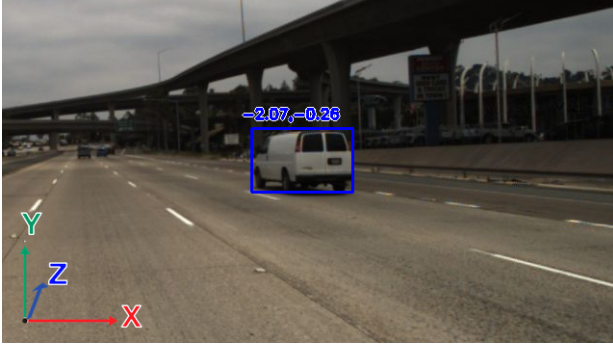


Fig. 1. Sample sequences from the TuSimple dataset at the last frame with their respective bounding box and velocity annotation in  $Z$  and  $X$  direction.

## II. RELATED WORK

**a) Depth and ego-motion estimation:** In recent years numerous approaches to monocular depth and ego-motion estimation have been explored thanks to the expressive power of convolutional neural networks. Supervised approaches [4], [25], [28] depend on pixel-wise depth annotations to be available for each pixel of the training set, which results in costly data collection. Unsupervised methods [17], [30] on the other hand depend heavily on camera intrinsics and rely on camera motion between successive frames to produce coarse relative depths, which are difficult to use for predicting exact distances.

**b) Velocity estimation:** Classical vision techniques for motion detection from a moving camera such as Yamaguchi *et al.*[29] first match points in successive frames and then filter them based on their location and compatibility with the epipolar geometry. A similar approach is used in Fanani *et al.*[5], where candidate objects are first filtered by a CNN which detects vehicles and then tests based on the epipolar geometry are used to determine whether these vehicles are moving or not.

The above approaches aim to only differentiate between static and moving vehicles with no indication of their velocity. Kampelmühler *et al.*[15], the winner of CVPR 2017 Vehicle Velocity Estimation Challenge [1], extends these approaches to predict the relative velocity of the vehicles in view of the ego-vehicle. This is achieved by passing the sequence of images through a depth [7] and flow [11] estimation network, as well as applying a classical tracking system [13]. The features extracted from these three sub-components are then concatenated and fed to a small neural network which predicts the velocity (see fig. 2 top).

**c) Tracking:** Object tracking is a classical computer vision problem [20]. The multiple instance learning tracker [2] expresses the problem as a classification task, where detections in previous frames are used as training data for future frames. MedianFlow [13] uses the forward-backward error to validate which points are robust predictors of object movement. The method is further improved in the TLD tracker [14] by disabling online learning when the object is occluded and by allowing the algorithm to re-

detect the object once it appears again. More recently, with the emergence of deep learning, specialized networks have been trained to explicitly track location of objects in video sequences [10]. Similarly, Siamese CNNs [3], [26] have been exploited to build a powerful embedding to discriminate whether an image patch contains the same object or not, therefore tracking objects by appearance similarity. For a systematic evaluation of tracking algorithms, we refer the reader to the VOT challenge results [18].

**d) Synthetic training data:** Synthetic training data have been successfully applied in various domains of computer vision, ranging from scene text detection [8] to optical flow estimation [11]. In the autonomous driving domain, the most widely used synthetic dataset is Virtual KITTI [6], which contains 50 photo-realistic videos generated by the Unity game engine. Thanks to the synthetic source of the data, the dataset comes with pixel-level annotations for segmentation, optical flow and depth, which would be virtually impossible and very expensive to achieve in real data. More recently, the SYNTHIA [22] and Synscapes [27] artificial driving datasets have also been introduced.

The main difference to our work is that the above datasets are based on photo-realistic representation of the world, whereas our mid-level representation consists of mere bounding boxes, which are significantly easier to generate.

## III. METHOD

Our goal is to estimate the velocity of other vehicles imaged from a camera rigidly mounted on the ego-vehicle and looking forward. We can model this situation as follows. The input to the model is a sequence  $\mathbf{I} = (I_1, \dots, I_T)$  of  $T$  video frames extracted from the camera. We also assume to have a bounding box  $b_T$  tightly enclosing the vehicle of interest at the end of the sequence  $T$ . The output of the model  $\Phi$  is a 2D vector  $V = \Phi(\mathbf{I}, b_T) \in \mathbb{R}^2$  representing the velocity of the target vehicle at time  $T$  projected on the ground plane relative to the ego-vehicle.<sup>1</sup>

### A. Geometry and elementary velocity estimation

Next, we describe in some detail the geometry of the problem and provide a naïve solution based solely on pro-

<sup>1</sup>Estimating the vertical velocity  $Y$  is essentially irrelevant for this application.



Fig. 2. The state-of-the-art architecture of [15] (top) consists of three different sub-networks and separate models for each of the three distance intervals. Our architecture (bottom) trained on synthetic data is significantly more straightforward, uses a single model and achieves comparable accuracy.

jecting bounding boxes into 3D world coordinate system. The physical constraint of the setup results in several simplifications compared to the general imaging scenario. Let  $P = (X, Y, Z) \in \mathbb{R}^3$  be a 3D point expressed in the ego-vehicle reference frame. We can assume that the camera is at a fixed height  $H$  from the ground looking straight ahead. Hence, point  $P$  projects to the image point  $p = (u, v)$ ,  $u = f \frac{X}{Z}$ ,  $v = f \frac{Y+H}{Z}$  where  $f$  is the focal length of the camera.<sup>2</sup>

Now assume that the bounding box  $b = (x, y, w, h) \in \mathbb{R}^4$  is given by the image coordinates  $(u, v) = (x + w/2, y + h)$  of the mid point of the bottom edge of the box and by the box width and height  $(w, h)$ . To a first approximation, we can assume that  $(u, v)$  is the image of a certain virtual 3D point  $P = (X, 0, Z)$  rigidly attached to the vehicle at ground level. Hence, since we know the height  $H$  of the camera, we can readily infer the depth or distance  $Z$  of the vehicle, and hence the 3D point, as

$$Z = fH \frac{1}{v}, \quad X = H \frac{u}{v}. \quad (1)$$

If we can track the bounding box  $b_t$  over time  $t \in [1, T]$ , then we can use eq. (1) to estimate the coordinates  $P_t = (X_t, 0, Z_t)$  of the 3D point relative to the ego-vehicle and obtain the velocity as the derivative  $V = (\dot{X}_T, \dot{Z}_T)$ . However, owing to the varying quality of road surfaces and inclines or declines along a road this technique provides very poor estimate of vehicle velocity, especially for vehicles which are further away (see table II). Indeed, analyzing eq. (1) and assuming a camera with standard image resolution, we come to a conclusion that for vehicles in distance  $d > 20\text{m}$  the approach requires sub-pixel accuracy of the bounding-box estimate for a reasonable estimate of 3D position and hence velocity, which simply is not realistic.

### B. Deep learning for velocity estimation

Sophisticated models which combine several streams of information from image pixels (depth, optical flow) perform significantly better, so one may be tempted to ascribe the poor performance of the geometric approach to the fact that too much information is discarded by looking at bounding

boxes only. We show that, somewhat surprising, *this is not the case*. Instead, we show below that bounding boxes *are* sufficient provided that the modelling of dynamics is less naïve.

**a) State-of-the art baseline:** Our reference model is the one of Kampelmühler *et al.* [15]. They propose a complex network that combines three data streams, all derived from the video sequence  $\mathbf{I}$  (see fig. 2 top). One stream applies a pre-trained monocular depth-estimation network, called MonoDepth [7], to estimate a 3D depth map from the video. Another stream applies FlowNet2 [11], a state-of-the-art optical flow estimation network, to estimate instead optical flow. The last stream uses an off-the-shelf tracker [13] to track the bounding box  $b_T$  backward through time, and thus obtain a simplified representation  $(b_1, \dots, b_T)$  of the vehicle trajectory in the image. The output of the different streams are concatenated and passed through a multi-layer perceptron (MLP) to produce the final velocity estimate  $V = \Phi(\mathbf{I}, b_T)$ . In addition to pre-training MonoDepth, FlowNet2, and the tracker on external data-sources, the MLP, fusing the information, is trained on an ad-hoc benchmark dataset that contains vehicle bounding boxes for one frame with their ground-truth velocities and positions measured via a LiDAR (section IV).

Furthermore, three individual models of different-sized MLPs are used to determine velocity within the different vehicle distance ranges used in the evaluation. These range from 3 layers of 40 hidden neurons to 4 layers of 70 hidden neurons with increasing model size for vehicles further away [15]. The final velocity prediction is given by averaging output of 5 model instances, therefore in total 15 distance-specific models are used at testing time.

**b) Our Model:** Similar to [15] we use a fully connected neural network with 4 hidden layers of 70 neurons each with CReLU[23] activation and Dropout[24] between layers during training. This allows us to reduce the runtime of our method significantly as we do not require estimation of optical flow and depth which allows our runtime to reduce from 423ms to 10ms and only one model is trained reducing the inference time further.

**c) Estimating velocity from bounding boxes:** Next, we describe our architecture to estimate vehicle velocity (see fig. 2 right). We consider an input video sequence  $\mathbf{I} = (I_1, \dots, I_T)$  of  $T$  video frames capturing a vehicle at

<sup>2</sup>The image coordinates  $(u, v)$  are standardized, with  $(0, 0)$  corresponding to the view direction, and are in practice related to pixels coordinates via a non-linear transformation that accounts for the camera intrinsic parameters, including effects such as radial distortion. We assume that the intrinsic parameters are known and that their effects has already been removed from the data.

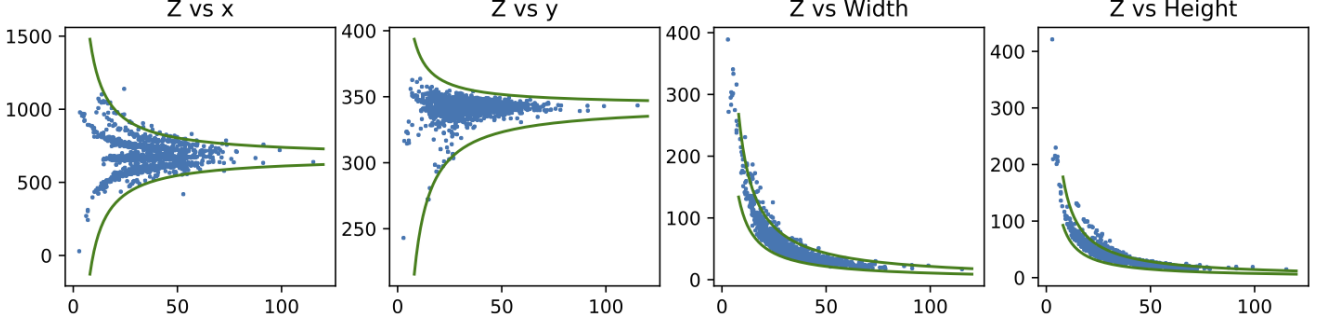


Fig. 3. **Vehicle statistics.** Top left: coordinate of the left edge of the bounding box image location vs the depth  $Z$  of the enclosed vehicle. This is constrained by green lines that represent the pixel coordinates of 3D points 9m left and right of the camera and placed at corresponding depths. Other plots in order: coordinate  $y$  (constraints at  $[0, 2.5\text{m}]$ ), box height  $h$  ( $[1.3\text{m}, 2.5\text{m}]$ ) and box width  $w$  ( $[1.5\text{m}, 3\text{m}]$ ).

time  $t \in [1, T]$ . As shown above, in the reference model, the velocity estimate  $V$  is then obtained as

$$V = \Phi(\mathbf{I}, b_T) \quad (2)$$

where  $\Phi$  is neural network or a combination of several neural networks as in [15], taking video sequence  $\mathbf{I}$  as an input and a bounding box  $b_T$  in the last frame.

In our paper, we decompose  $\Phi$  into two mappings  $\Psi$  and  $\Xi$

$$V = \Psi(\Xi(\mathbf{I}, b_T)) \quad (3)$$

by introducing an intermediate representation  $\mathbf{b} = (b_1, \dots, b_T)$  representing the vehicle image location at time  $t \in [1, T]$   $\mathbf{b} = \Xi(\mathbf{I}, b_T)$ ,  $V = \Psi(\mathbf{b})$ , where  $\Xi$  is a off-the-shelf tracker component [13] and  $\Psi$  is the vehicle velocity estimator we train.

Using the above decomposition, we only require  $(\mathbf{b}, V)$  as supervision pairs, and not the  $(\mathbf{I}, V)$  pairs as in the original formulation [15], which are extremely expensive to obtain as it implies capturing and annotating many driving video sequences.

As before, bounding boxes  $\mathbf{b}$  are represented as quadruples  $(x, y, w, h)$ . The trajectories  $\mathbf{b}$  are passed to a filter  $g(\mathbf{b})$  that applies temporal Gaussian smoothing to each coordinate independently. Finally, the output of the filter is flattened to a  $4T$ -dimensional vector and fed to a multi-layer perceptron  $\bar{\Psi} : \mathbb{R}^{4T} \rightarrow \mathbb{R}$ . Hence, the overall model at inference time can be written as  $V = \Psi(\mathbf{b}) = \bar{\Psi}(\text{vec}(g(\Xi(\mathbf{I}, b_T))))$ .

At training time, assuming the tracker  $\Xi$  is fixed, we however still need to train  $\Psi$ , using the pairs  $(\mathbf{b}, V)$  by minimizing the loss  $\|V - \Psi(\mathbf{b})\|^2$ . Next we show how the pairs can be obtained.

#### IV. LEARNING FROM REAL OR SYNTHETIC DATA

We begin by discussing a real benchmark dataset sufficient to train our model and then we discuss how we can generate analogous synthetic data effectively in the space of object bounding boxes.

##### A. Real data: TuSimple

While many driving datasets exist in the realm of autonomous driving many focus on single frame tasks such as object detection in 2D/3D, depth estimation, semantic

segmentation and localisation. The only dataset with necessary annotations for our task with comparable work come from TuSimple[1]. The TuSimple dataset was introduced for the CVPR2017 Autonomous Driving velocity estimation challenge [1]. It contains 1074 (269) driving sequences with 1442 (375) annotated vehicles in the training (respectively testing) set, split into three subsets based on the distance between the ego-vehicle and the observed car (see table I). The dataset was recorded on a motorway by an standard camera (image resolution  $1280 \times 720$ ) mounted on the roof of the ego-vehicle. Each driving sequence contains 40 frames, captured at 20fps as seen in fig. 1. LiDAR and radar were simultaneously used to capture the position and velocity of nearby vehicles and recorded data were then used to give the ground truth velocity and 3D position of each vehicle in the last frame of the sequence. Additionally, a manually created bounding box (in the image space) is available in the last frame for each vehicle. Using the notation from section III-B, the dataset therefore provides 1442 training and 375 testing triplets  $(\mathbf{I}, b_T, V)$ . As shown in fig. 4 the TuSimple dataset spans a wide range of distances from the camera with a high bias towards the same lane as is typical in driving imagery. Velocity in both  $Z$  and  $X$  is roughly uniformly distributed around 0 which is expected in motorway situations.

##### B. Synthetic data

By distilling the information contained in an image to bounding boxes  $\mathbf{b}$  we now have data which is highly-interpretable and easy to characterise statistically; the latter can be used to generate *synthetic* training data, which ideally will have very similar statistical properties to the real data. We could also choose to simulate vehicles with speeds not normally seen in the real world to train a network capable of understanding such abnormal situations.

	Near $d < 20$	Medium $20 < d < 45$	Far $d > 45$	Total
Train	166	943	333	1442
Test	29	247	99	375

TABLE I

THE NUMBER OF ANNOTATED VEHICLES IN THE TUSIMPLE DATASET [1] AND THEIR SPLIT BASED ON THEIR DISTANCE FROM THE EGO-VEHICLE.



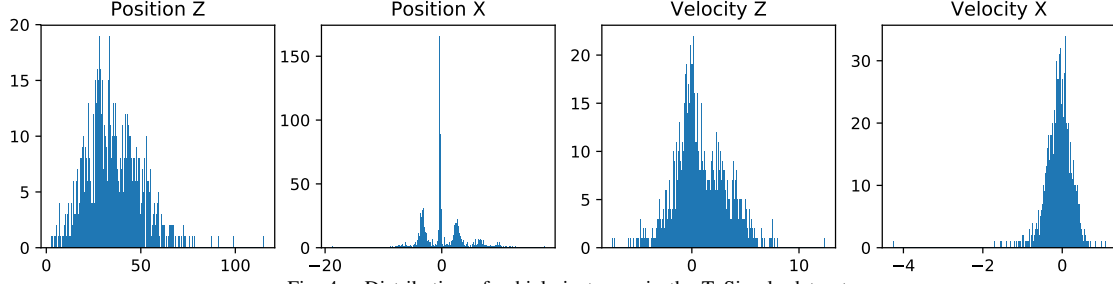


Fig. 4. Distribution of vehicle instances in the TuSimple dataset.

We use the training subset of the TuSimple dataset (see section IV) to infer these statistical properties. In fig. 3-left we plot the bounding box horizontal coordinates  $x$  vs the depth of the enclosed vehicle. There is some obvious structure in the data; in particular, several lanes to the left and to the right of the ego-vehicle are clearly visible. The other plots show the  $y$  coordinate as well as the bounding box width and height. The latter are highly constrained by the physical sizes of vehicles.

We found learning to be sensitive to the distribution of vehicles locations and less so of car velocities and sizes. We thus represent the location distribution empirically (fig. 5) and sample from TuSimple the first bounding box in each to obtain its 3D ground point  $(X, 0, Z)$ . This is then re-projected to the image as explained in section III-A to obtain the bounding box location  $(x, y)$ . The box height and width are obtained indexing with the depth  $Z$  polynomial fits to the TuSimple height/width data (fig. 3). Finally, the track is simulated by sampling a velocity vector  $V$  from a Gaussian fit of the TuSimple velocity data (fig. 4) and integrating the motion (fig. 6).

Note that only the bounding box positions are empirical, whereas the other parameters are sampled from very simple distributions. We show that this is sufficient to train models that achieve excellent performance on real data. This helps isolating what are the important/sensitive priors (location) and what are not (size, velocity) for velocity estimation.

## V. EVALUATION

We evaluated our approach using two models. For both models, we used MLP with 4 hidden layers, trained for

150 epochs, using dropout of 0.2 after the Concatenated ReLU[23] activation function, learning rate with learning rate  $6 \times 10^{-4}$  adjusted using exponential decay with decay rate set to 0.99.

In the first model denoted as *MLP-tracker*, we initially processed all video frames by the MedianFlow [13] tracker, using the bounding box  $b_T$  from the ground truth for tracker initialization. We then used the resulting sequences  $\mathbf{b} = (b_1, \dots, b_T)$  as the training data for the MLP (see section III-B). At test time, we followed the same procedure, which is pre-processing the input video sequence by the tracker, and then feeding the intermediate representation  $\mathbf{b}$  into the MLP to obtain the velocity estimate.

We follow the TuSimple competition protocol, and report velocity estimation accuracy  $E_v$  calculated as the average over the three distance-based subsets (table I):

$$E_v = \frac{E_v^{\text{near}} + E_v^{\text{medium}} + E_v^{\text{far}}}{3}, \quad E_v^S = \frac{1}{|S|} \sum_{i \in S} \|V_i - \hat{V}_i\|^2, \quad (4)$$

where  $V_i$  is the ground truth velocity from the LiDAR sensor and  $\hat{V}_i$  is method's velocity estimate. We also note that according to the dataset authors, the overall ground-truth accuracy is at around 0.71m/s, however the accuracy will almost certainly depend on the distance of the observed car.

Using the above model, we reach the overall velocity estimation error of 1.29, which outperforms the state-of-the-art<sup>3</sup>

<sup>3</sup>Kampelmühler *et al.* also report the error of 1.86 and 1.25 for their method using only tracking information, however this was not the competition entry and it is not obvious from the paper how the latter number was reached and what data were used for training.

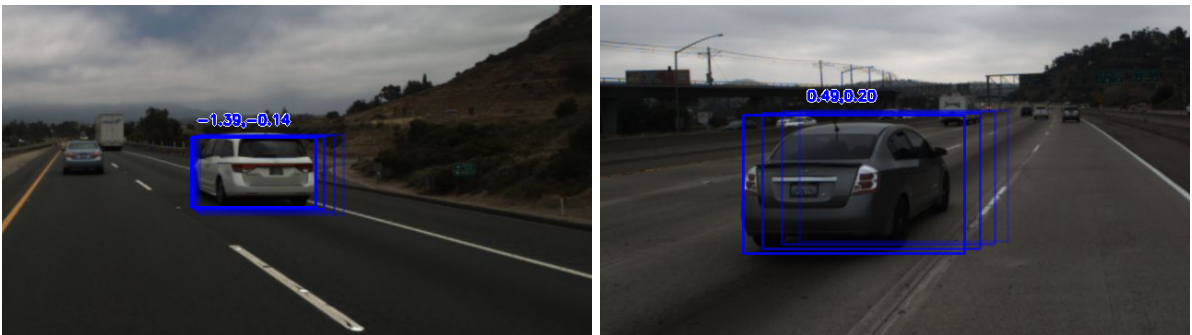


Fig. 5. Here we show an example of our synthetic boxes in these cases generated starting at the same location as a real object. The fainter boxes depict the location of the object in subsequent frames for the velocity written above which may differ from the actual vehicles path in real data

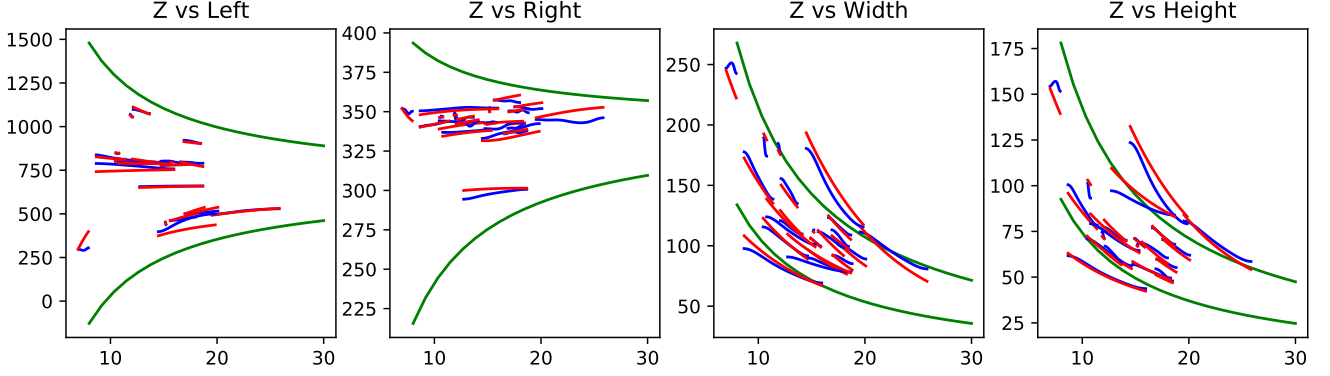


Fig. 6. **Real vs synthetic vehicle sequences.** The motion of the bounding box in the video sequence - vehicle motion in real video as captured by the tracker (blue), and synthetic motion generated by our method (red). In the width and height plot we see some tracking inaccuracy when the blue lines increase going to the right.

and significantly more complex method of Kampelmühler *et al.*[15]. This experiment also shows, that the intermediate representation **b** contains sufficient amount of information to successfully infer vehicle velocity.

In the second model *MLP-synthetic*, we instead used the synthetic training data as the intermediate representation **b**. We generated 11536 samples using the generation procedure described in section IV-B and used them as the training samples for the MLP. At testing time, we again used the MedianFlow tracker to get the intermediate representation **b** from real video sequences and fed them into the MLP to obtain vehicle velocity estimates (see fig. 2 bottom). Because our synthetic boxes are generated without noise inherently coming from the tracker road surface and other sources, at test time we apply a Gaussian filter with  $\sigma = 5$  in each coordinate of the bounding box as pre-processing to remove noise.

The resulting model has the velocity estimation error of 1.28, with the biggest improvement in the Far range (see table II). Generating more synthetic data for training did not result in further improvement in accuracy, which we contribute to the relatively small size of the testing set and the relative low variance of vehicle speed owing to motorway driving style.

## VI. CONCLUSION

In this paper, we showed how to efficiently predict the velocity of other vehicles based only on a video from a standard monocular camera by introducing an intermediate representation which decouples perception from velocity estimation. We also showed how priors in real data can be exploited to generate synthetic data in the intermediate representation and that such synthetic training data can be used to build a system capable of processing real-world data without any loss in accuracy.

The decomposition is advantageous not only in terms of the ability to easily generate training data with the desired parameters, but also to model different driving styles or various emergency situations. Last but not least, it also ensures each component is individually verifiable, which

	$E_v$	$E_v^{\text{near}}$	$E_v^{\text{medium}}$	$E_v^{\text{far}}$
Kampelmühler <i>et al.</i> [15] (1st)	1.30	0.18	<b>0.66</b>	3.07
Wrona (2nd)	1.50	0.25	0.75	3.5
Liu (3rd)	2.90	0.55	2.21	5.94
geometric reprojection	8.5	0.48	1.50	23.60
Ours (MLP-tracker)	1.29	0.18	0.70	2.99
Ours (MLP-synthetic)	<b>1.28</b>	<b>0.17</b>	0.72	<b>2.96</b>
<i>LiDAR</i>	0.71			

TABLE II

VEHICLE VELOCITY ESTIMATION ERROR  $E_v$  ON THE TUSIMPLE DATASET, INCLUDING THE TOP 3 BEST-PERFORMING METHODS OF THE CVPR 2017 VEHICLE VELOCITY ESTIMATION CHALLENGE [1]. THE AVERAGE ACCURACY OF THE LiDAR SENSOR USED FOR GROUND TRUTH ACQUISITION ADDED FOR REFERENCE. RUNTIME OF KAMPELMÜHLER *et al.*[15] IS 423ms OWING TO THE VARYING AND COMPLEX INPUTS REQUIRED FOR A FORWARD PASS, OUR METHOD CAN PERFORM A FORWARD PASS IN 10ms ON THE SAME HARDWARE(RUNTIME FOR OTHER METHODS ARE NOT PUBLIC).

is essential to meet reliability standards for autonomous driving.

## ACKNOWLEDGEMENT

We are very grateful to Continental Corporation for sponsoring this research.

## REFERENCES

- [1] CVPR 2017 workshop on autonomous driving. <https://github.com/TuSimple/tusimple-benchmark>. Accessed: 2020-02-15.
- [2] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 983–990. IEEE, 2009.
- [3] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.
- [4] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.



- [5] Nolang Fanani, Matthias Ochs, Alina Sturck, and Rudolf Mester. CNN-based multi-frame IMO detection from a monocular camera. In *2018 IEEE Intelligent Vehicles Symposium, IV 2018, Changshu, Suzhou, China, June 26-30, 2018*, pages 957–964, 2018.
- [6] A Gaidon, Q Wang, Y Cabon, and E Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016.
- [7] Clément Godard, Oisín Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.
- [8] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2315–2324, 2016.
- [9] Sinan Hasirlioglu, Andreas Riener, Werner Huber, and Philipp Wintersberger. Effects of exhaust gases on laser scanner data quality at low ambient temperatures. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1708–1713. IEEE, 2017.
- [10] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.
- [11] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. *CoRR*, abs/1612.01925, 2016.
- [12] Road vehicles – functional safety. Standard, International Organization for Standardization, Geneva, Switzerland, 2011.
- [13] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *2010 20th International Conference on Pattern Recognition*, pages 2756–2759, Aug 2010.
- [14] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2012.
- [15] Moritz Kampelmühler, Michael G Müller, and Christoph Feichtenhofer. Camera-based vehicle velocity estimation from monocular video. *arXiv preprint arXiv:1802.07094*, 2018.
- [16] Dominik Kellner, Michael Barjenbruch, Klaus Dietmayer, Jens Klappstein, and Jürgen Dickmann. Instantaneous lateral velocity estimation of a vehicle using doppler radar. In *Proceedings of the 16th International Conference on Information Fusion*, pages 877–884. IEEE, 2013.
- [17] M. Klodt and A. Vedaldi. Supervising the new with the old: learning sfm from sfm. In *European Conference on Computer Vision*, 2018.
- [18] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Cehovin Zajc, Tomas Vojir, Gustav Hager, Alan Lukezic, Abdelrahman Eldesokey, et al. The visual object tracking vot2017 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1949–1972, 2017.
- [19] Matti Kuttila, Pasi Pykönen, Werner Ritter, Oliver Sawade, and Bernd Schäufele. Automotive lidar sensor development scenarios for harsh weather conditions. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 265–270. IEEE, 2016.
- [20] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.
- [21] Rob Palin, David Ward, Ibrahim Habli, and Roger Rivett. Iso 26262 safety cases: Compliance and assurance. 2011.
- [22] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [23] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. *CoRR*, abs/1603.05201, 2016.
- [24] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):19291958, January 2014.
- [25] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5038–5047, 2017.
- [26] Jack Valmadre, Luca Bertinetto, João Henriques, Andrea Vedaldi, and Philip HS Torr. End-to-end representation learning for correlation filter based tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2805–2813, 2017.
- [27] Magnus Wrenninge and Jonas Unger. Synscapes: A photorealistic synthetic dataset for street scene parsing. *arXiv preprint arXiv:1810.08705*, 2018.
- [28] Dan Xu, Elisa Ricci, Wanli Ouyang, Xiaogang Wang, and Nicu 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*, pages 5354–5362, 2017.
- [29] Koichiro Yamaguchi, Takeo Kato, and Yoshiki Ninomiya. Vehicle ego-motion estimation and moving object detection using a monocular camera. In *18th International Conference on Pattern Recognition (ICPR’06)*, pages 610–613. IEEE, 2006.
- [30] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1858, 2017.

# Part II

## Monocular Depth Estimation

## Chapter 4

# Monocular Depth Estimation with self-supervised instance adaption

# Monocular Depth Estimation with Self-supervised Instance Adaptation

Robert McCraith, Lukas Neumann, Andrew Zisserman, and Andrea Vedaldi \*

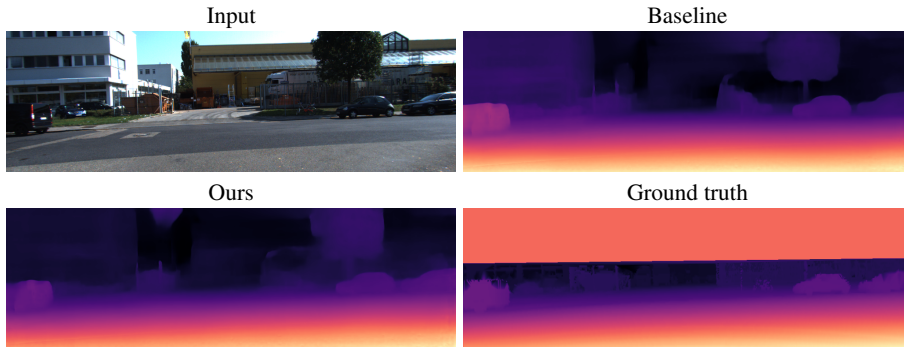
University of Oxford

**Abstract.** Recent advances in self-supervised learning have demonstrated that it is possible to learn accurate monocular depth reconstruction from raw video data, without using any 3D ground truth for supervision. However, in robotics applications, multiple views of a scene may or may not be available, depending on the actions of the robot, switching between monocular and multi-view reconstruction. To address this mixed setting, we proposed a new approach that extends any off-the-shelf self-supervised monocular depth reconstruction system to use more than one image at test time. Our method builds on a standard prior learned to perform monocular reconstruction, but uses self-supervision at test time to further improve the reconstruction accuracy when multiple images are available. When used to update the correct components of the model, this approach is highly-effective. On the standard KITTI benchmark, our self-supervised method consistently outperforms all the previous methods with an average 25% reduction in absolute error for the three common setups (monocular, stereo and monocular+stereo), and comes very close in accuracy when compared to the fully-supervised state-of-the-art methods.

## 1 Introduction

Using images to reconstruct the 3D geometry of an environment, for example in the form of a depth map, has important applicators in robotics, autonomous driving, augmented reality, scene understanding and more [2–6]. Since inferring depth from a single image is

\* Authors are with Visual Geometry Group, Dept. of Engineering Science, University of Oxford  
{robert, lukas, az, vedaldi}@robots.ox.ac.uk



**Fig. 1.** Qualitative samples from the KITTI dataset [1]. Self-supervised instance adaptation during inference consistently improves depth estimation across different challenging scenarios.

ill-posed, traditional implementations use multiple views, obtained either from a moving camera or from a multi-camera setup (usually a stereo pair). Yet, as humans, we are able to navigate an environment or even drive with a single eye [7]. In general, we can perform *monocular depth prediction*, estimating depth from a single 2D image.

Monocular depth prediction cannot be obtained via visual geometry alone. Instead, the problem can only be solved by using a powerful prior on the possible 3D shape and appearance of objects and scenes. Such a prior must necessarily be *learned from data*, which explains why humans are good at this task. However, with modern deep learning techniques, it is now possible to learn effective 3D priors for use in algorithms. Even more remarkably, recent *self-supervised* approaches [3, 8–10] have shown that 3D priors can be learned with no access to 3D ground-truth nor multiple camera setups, using only *videos shot from a single camera*. This avoids ad-hoc setups involving additional modalities such as LiDARs to collect ground-truth 3D data for training.

In this paper, we observe that, since self-supervised monocular depth estimation learns from the consistency between subsequent video frames [3] with no need for any external ground-truth signal, the *same principle can be exploited during inference* in addition to just during training. We show that, given an existing monocular depth prediction model and a careful selection of which components to optimize, we can update certain model parameters (weights) at inference time through standard back-propagation. Our method can seamlessly be extended to mixed monocular and multi-view depth estimation. We also show that we can continually update the weights through a sequence for greater performance.

This idea allows to use the strong monocular depth prediction prior learned by the model during training to robustly interpret multiple views at test time, improving individual depth predictions. Furthermore, this refinement process can also be applied *continually* during inference, which not only improves the run-time of the method but also is a form of domain self-adaptation: for example, in a driving scenario, this allows a network trained on a dataset representative of one location to be deployed in areas which look different, automatically fine-tuning the model. We want to emphasize, that this *does not imply training on testing data* in the usual supervised learning sense, which would of course be invalid; the reason is that there is no ground truth or additional supervision involved, but rather we simply perform an additional post-processing step at inference time for each sample.

We show empirically that our depth estimation method significantly outperforms all other self-supervised depth estimation baselines, without requiring any re-training, and very nearly matches the quality of fully-supervised systems. We also significantly outperform recent methods which observe and aggregate information from numerous frames at test time.

We also believe this idea could be further extended in several ways. First, application of our method is optional and can be engaged only when motion is observed, reverting back to standard monocular depth estimation otherwise. Second, we show that it can benefit from multiple sensors such as stereo pairs that have also been used for self-supervision [8]. Third, it can likely be extended to many other scenarios where self-supervision is applicable, to improve inference or perform self-adaptation at test time.

## 2 Previous Work

### 2.1 Other Sensors

From practical perspective, depth in a scene can be estimated using additional sensors such as LiDARs or radars, but these modalities have their own limitations: LiDARs have problems in certain weather conditions [11], they only produce sparse depth maps which may not be enough for far-away or small objects and they have lower refresh rate when compared to many other sensors (10–15Hz [12]). Similarly, radars suffer from reflections and interference and they struggle to detect small or slowly-moving objects [13]. Stereo vision systems are very sensitive to precise calibration and could face problems with misalignment of cameras following even a negligible collision, but the main drawback is the lack of redundancy — a moving autonomous vehicle or a robot in urban environment cannot simply stop working if for example one of the cameras becomes occluded by dirt, as that would be potentially very unsafe. By uniting monocular and multi-view reconstructions setups in a robust manner, our method can help to address these challenges.

### 2.2 Supervised Depth

Owing to the scale ambiguity present when capturing images with a single camera the prediction of depth from a single image is ill-posed as the same image can result from scenes at various scales. Early approaches to this problem revolved around the use of geometry to extract point to point correspondences between images and triangulating to estimate depth [14, 15]. Through the use of deep learning solutions to this problem have emerged which utilise additional inputs for supervision typically in the form of LiDAR, RGB-D cameras and stereo pairs. This allows a model to be trained to exploit the relationship between colour images and their corresponding depth.

Eigen et al. [16] applied multi-scale networks to refine estimated depth maps from low spatial resolution to high resolution depth maps independently. Xie et al. [17] used skip-connections to fuse low resolution information dense layers with higher resolution information sparse earlier layers to refine predictions further. [18–20] apply multi-layer deconvolutional networks to recover coarse-to-fine depth while others rely on conditional random fields to improve fine details [21, 22]. DORN [4] introduced space-increasing discretization to transform depth prediction from a regression problem to one of ordinal regression, encouraging faster convergence and improving accuracy. Lee et al. [23] utilise local planar guidance layers at multiple stages in the decoder to improve performance.

### 2.3 Self-supervised Depth

Even with the use of additional sensors capturing ground truth data is not without difficulty. LiDAR produces sparse point clouds, using RGB-D cameras in bright environments results in unstable depth maps and stereo setups require precise calibration and lose accuracy quickly as distance increases. With these problems in mind there has been a recent interest in unsupervised methods for depth map prediction. Early works in this area focused on using stereo images in training such as [17] which proposed prediction

of discretized depth for novel view synthesis for VR and 3D video applications, [19] extended this approach to continuous values and Monodepth [8] added a left-right depth consistency term to the loss to further improve results. This concept was generalized to remove the need for stereo pairs and instead predict the pose between two sequential frames in a video during training as seen in SfMLearner [3].

Unlike the stereo version where a scene is captured from two viewpoints simultaneously with a sequence of frames, non-stationary objects can move causing problems when comparing an image to another warped to have the same viewpoint, resulting in many methods employing another output mask to discount these regions. Caser et al. [24, 25] explicitly model the 3D motion of objects in the scene, which allows their method to learn from dynamic scenes with lot of motion. They also employ a certain test-time refinement step, but unlike our method they to first infer all test images and then run the refinement of the whole set which makes it very impractical, they use an ad-hoc heuristics to determine whether to use the refined output or not and they optimize parameters of the whole network which we found sub-optimal.

In Monodepth V2 [9] this problem is tackled using a per pixel minimum re-projection loss which aims to exclude pixels that have become occluded frames surrounding the current frame which would normally results in large error which effects the average error computed for this location. Yin et al. [26] on the other hand decompose the image into rigid and non-rigid components using flow or semantic maps to reduce this re-projection error. GLNet [27] introduce a new loss to capture geometric constraints and predict camera intrinsics, and they also propose several test-time refinement strategies. Their approach is however approx. 20 times slower than ours and less accurate (see table 5), despite using additional loss terms beyond photometric similarity which increase computation time and complexity - our system only requires a basic calculation of L1 and SSIM error resulting in faster more robust adaption when we only train the encoders to capture the domain shift of training to real data.

Last but not least, Patil et al. [28] train an LSTM to specifically aggregate information across multiple frames - in contrast, our method does not require such additional components, and is rather able to continuously update its parameters through a video sequence, achieving higher accuracy at much faster run time.

### 3 Method

In this section, we describe the architecture of our network and the self-supervised loss function used in training and in inference. We then show how to update certain parameters (weights) of a trained model dynamically at run time for each image instance to optimize depth prediction for each specific image instance individually. This strategy helps us overcome the issue of poor generalisation to natural augmentations [29, 30] and is possible owing to the availability of temporally adjacent frames in the case of monocular training and stereo pairs of images in car setups with such equipment in real world scenarios.

### 3.1 Self-supervised Depth Estimation

We base our method off of [9] and adopt the network architecture, and loss functions of this work to focus solely on the test time adaption process. The following describes the approach taken in [9] which we replicate as our baseline. We formulate the depth estimation problem as novel view synthesis, where a network  $\Psi$  produces two auxiliary variables — a dense depth map  $D$  and a pose estimate  $T$ . Given a pair of input images (two subsequent frames in a video, left & right images from a stereo camera, etc.), the auxiliary variables  $D, T$  are then used to warp one of the images into another, enforcing visual consistency between the warped and the other image.

More formally, and assuming two sequential images  $I^t$  and  $I^{t+1}$  as an input, the network  $\Psi$  is trained to produce estimates  $D_t$  and  $T_t$  to minimize the appearance loss between the original  $I^t$  and the synthesized image  $\hat{I}^t = \mathcal{W}(I^{t+1}, D_t, T_t, K)$ , where  $\mathcal{W}$  is the warping operation [31] and  $K$  is the camera intrinsics assumed to be known. We thus have:

$$\mathcal{L}(I^t, I^{t+1}) = \alpha E_p(I^t, \hat{I}^t) + E_{\text{dis}}(I^t, \hat{I}^t) \quad (1)$$

$$\hat{I}^t = \mathcal{W}(I^{t+1}; D_t, T_t, K) \quad (2)$$

$$D_t, T_t = \Psi(I^t, I^{t+1}) \quad (3)$$

where, for training, the network  $\Psi$  is optimized using the loss function eq. (1) using the standard back-propagation, as the whole system is fully differentiable.

Similarly to [9], the loss function consists of a photometric loss term  $E_p$ , which is a combination of  $L^1$  and SSIM [32, 33] loss, and a disparity smoothness loss  $E_{\text{dis}}$  [8]. The photometric loss  $E_p$  also incorporates the auto-masking term  $\mu_{ij}$  [9], which reduces the error when a patch becomes occluded or where a patch is no longer visible due to camera motion. For our use case this component ensures that when the ego vehicle is stationary our depth predictions don't change as the entire image is masked from the loss term, other works [3, 8] used a predictive mask which could also be used for this purpose, Monodept V2 [9] however showed this method to be superior.

$$E_p(I^t, \hat{I}^t) = \frac{1}{N} \sum_{i,j} \mu_{ij} P(I_{ij}^t, \hat{I}_{ij}^t) \quad (4)$$

$$P(p, q) = \beta \frac{1 - \text{SSIM}(p, q)}{2} + (1 - \beta) \|p - q\|^1 \quad (5)$$

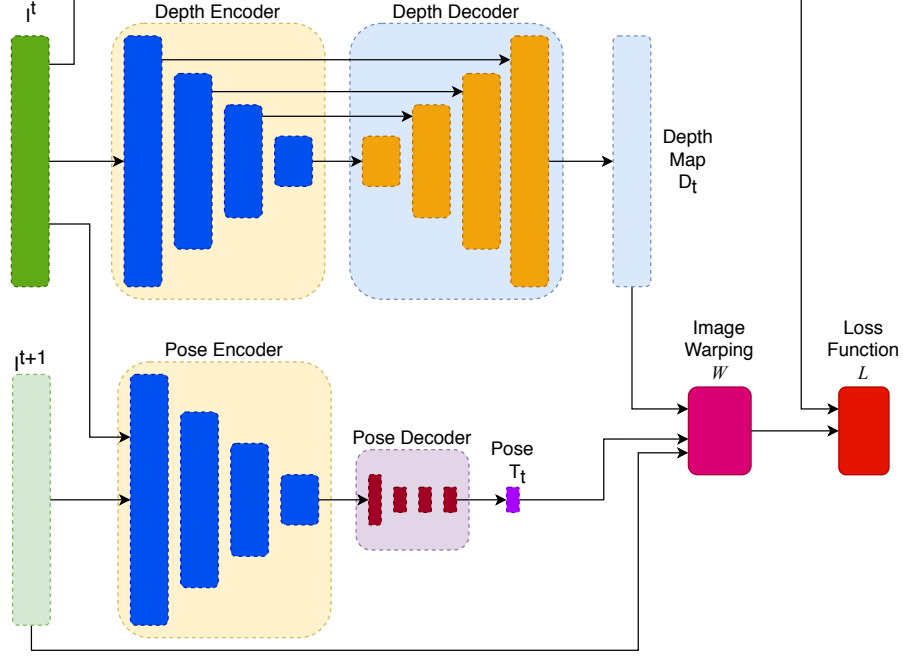
$$\mu_{ij} = [P(I_{ij}^t, I_{ij}^{t+1}) < P(I_{ij}^t, \hat{I}_{ij}^t)] \quad (6)$$

where  $[\cdot]$  indicates the Iverson bracket and consistently with the literature we set  $\alpha = 1$  and  $\beta = 0.85$ .

### 3.2 Instance Adaptation at Inference

We now introduce our main idea. Thanks to the *self-supervised formulation*, the loss function eq. (1) does not depend on any ground truth, but only on the observed images,





**Fig. 2.** Monocular depth estimation pipeline [9]. Modules enabled for instance adaptation during inference marked with yellow background

and can thus be used at *test time* to further improve the model by minimizing the same loss function through standard back-propagation for a defined number of steps. As a result, we get a depth estimate which is optimized for each specific observation instance (pair of images) individually.

Note that this *does not* imply training on testing data in the usual supervised learning sense, which would of course be invalid; the reason is that there is no ground truth or additional supervision involved, but rather we simply perform an additional post-processing step at inference time for each sample. This idea is inspired by 3D human pose literature [34], where a 3D model is fitted to 2D detections at run time.

Generalizing, we explore two basic scenarios: In the first scenario (**instance-wise**), we process each temporarily adjacent pair of images individually without any temporal consistency between different pairs, always starting from the same model weights obtained during the training process of the baseline model.

In the second scenario (**sequential**), we exploit the fact video frames have spatial and temporal consistency. We start the inference for the first image in a sequence with the trained weights as in the previous case, but we then keep updating the model weights through out the whole video. This enables the model to aggregate information across multiple frames and it should also be faster as we shouldn't have to make as many updates for one individual frame as in the previous case. This scenario also better reflects

Depth Encoder	Depth Decoder	Pose Encoder	Pose Decoder	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
baseline [9] (no instance adaptation)				0.115	0.898	4.728	0.190	0.879	0.961	0.982
whole network				0.207	3.152	8.008	0.272	0.729	0.891	0.951
✓				0.099	0.797	4.513	0.179	0.901	0.963	0.982
	✓			0.156	1.572	6.379	0.235	0.797	0.932	0.972
		✓		0.115	0.905	4.863	0.193	0.877	0.959	0.981
			✓	0.115	0.905	4.863	0.193	0.877	0.959	0.981
✓	✓			0.156	1.58	6.463	0.235	0.795	0.931	0.971
✓		✓		<b>0.097</b>	<b>0.788</b>	<b>4.464</b>	<b>0.178</b>	<b>0.905</b>	<b>0.964</b>	<b>0.982</b>
✓			✓	0.111	0.895	4.776	0.19	0.884	0.959	0.981
	✓	✓		0.151	1.443	6.217	0.229	0.806	0.936	0.973
	✓		✓	0.208	3.143	8.082	0.273	0.728	0.892	0.952
		✓	✓	0.115	0.905	4.863	0.193	0.877	0.959	0.981
✓	✓	✓		0.113	0.912	4.804	0.191	0.881	0.958	0.981
✓	✓		✓	0.212	3.319	8.183	0.276	0.725	0.887	0.948
✓		✓	✓	0.154	1.561	6.374	0.233	0.802	0.932	0.971
	✓	✓	✓	0.199	2.927	7.811	0.266	0.741	0.898	0.955
first layer				0.111	0.884	4.811	0.191	0.881	0.959	0.981
		first layer		0.115	0.905	4.863	0.193	0.877	0.959	0.981
last residual block				0.104	0.828	4.64	0.184	0.893	0.962	0.982
		last residual block		0.115	0.905	4.863	0.193	0.877	0.959	0.981
first layer		first layer		0.11	0.878	4.797	0.19	0.883	0.959	0.981
last residual block		last residual block		0.103	0.821	4.622	0.183	0.894	0.963	0.982

**Table 1.** The impact of enabling instance adaption for different network modules on the KITTI dataset [1]. For every configuration, we ran 10 SGD steps with  $LR=10^{-1}$  and BatchNorm updates disabled.

practical deployment in a moving vehicle, where frames are processed in a stream as they are captured, rather than independently in an undefined order.

### 3.3 Components and Parameterization

Note that the method of section 3.2 can also be interpreted as learning more information from a single image pair, or a single video sequence. An obvious concern is whether this might lead to *over-fitting* and poor performance. We found empirically that this is indeed the case — our method works well, but only provided that the adaptation is limited to the correct subset of model parameters (see table 1).

Concretely, consider the current state-of-the-art monocular depth estimation pipeline [9], shown in Figure 2. This model comprises four sub-networks: a depth encoder, a pose encoder, a depth decoder and a pose decoder. When minimizing eq. (1), we can optimize all sub-networks. However, the depth and pose decoders can also be interpreted as *parameterization* of depth and pose, expressing them as a function of corresponding codes computed by the encoders. We find that updating the encoders while keeping the decoders fixed works the best because the system can adapt to the visual appearance of the novel image, whilst keeping the prior knowledge about different object types static in the decoders. This and several other modelling choices are thoroughly assessed in section 4.3.

## 4 Experiments

After discussing the experimental data (section 4.1), we first optimize and ablate the various component of our approach (section 4.3), followed by a thorough evaluation of the the instance- and sequential-wise scenarios (sections 4.4 and 4.5).

## 4.1 Data

We evaluate our method using the standard KITTI 2015 Stereo dataset [1]. Consistently with the literature [8, 9, 24, 35], we use the data split of Eigen et al. [16], where for monocular setup we follow the pre-processing of Zhou et al. [3], which removes stationary scenes which would otherwise prove problematic for image warping based methods. This results in 39810 images with temporally adjacent frames for training, 4424 for validation and 697 for testing. In stereo and mixed monocular stereo set up, we use the full Eigen dataset, again consistently with the prior work. For monocular case, the method output is rescaled using the per-image median ground truth scaling as in [3, 9]; for the stereo case, absolute depth can be inferred directly using the camera baseline as a scaling factor.

## 4.2 Metrics

We follow the evaluation of previous works [8, 9, 16] and compute the following metrics:

- Abs Relative:  $\frac{1}{|D|} \sum_{y \in D} \frac{|y - y^*|}{y^*}$
- Square Relative:  $\frac{1}{|D|} \sum_{y \in D} \frac{\|y - y^*\|^2}{y^*}$
- Residual Mean Squared Error:  $\sqrt{\frac{1}{|D|} \sum_{y \in D} \|y - y^*\|^2}$
- Residual Mean Squared Error log:  $\sqrt{\frac{1}{|D|} \sum_{y \in D} \|\log y - \log y^*\|^2}$
- Threshold: % of  $y_i$  where  $\max(\frac{y_i}{y_i^*}, \frac{y_i^*}{y_i}) = \delta < threshold$

where  $y$  is the ground truth depth from LiDAR and  $y^*$  is our networks prediction for the same pixels the LiDAR point projects to in the input image. These are computed by scaling the predicted depth map to have the same median depth as the LiDAR points which project into the image, making the outputs of this network difficult to use for many downstream tasks, a problem addressed in the following chapter.

## 4.3 Optimal Configuration and Ablation

In order to find the optimal setting for our method, we validate which model components to adapt (section 4.3), how fast (section 4.3) and for how long (section 4.3). We also compare our method to a simple baseline in which depth and pose are optimized directly (section 4.3), which can also be interpreted as a special case of our approach.

**Components** As noted in section 3.3, the success of our method depends strongly on which parts of the model are optimized. This is assessed in table 1. As seen in the second row, simply updating all network parameters at inference time makes the accuracy significantly worse than the baseline (first line). We therefore systematically evaluate which sections of the network can have their weights updated without causing divergence. As described in section 3.3, our network can be split into four modules: two encoders based on a ResNet-18 [36] architecture and two decoders — one which takes

Learning Rate	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25^1$	$\delta < 1.25^2$	$\delta < 1.25^3$
baseline	0.115	0.905	4.863	0.193	0.877	0.959	0.981
$10^{-5}$	0.115	0.905	4.863	0.193	0.877	0.959	0.981
$10^{-4}$	0.114	0.904	4.861	0.193	0.878	0.959	0.981
$10^{-3}$	0.113	0.897	4.842	0.192	0.879	0.96	0.981
$10^{-2}$	0.107	0.852	4.719	0.186	0.889	0.961	0.981
$10^{-1}$	0.097	0.774	4.455	0.178	0.904	0.964	0.982
1	0.279	2.763	8.577	0.384	0.563	0.78	0.894
10	0.451	5.596	12.153	0.591	0.314	0.57	0.768

**Table 2.** Instance adaptation learning rate ablation

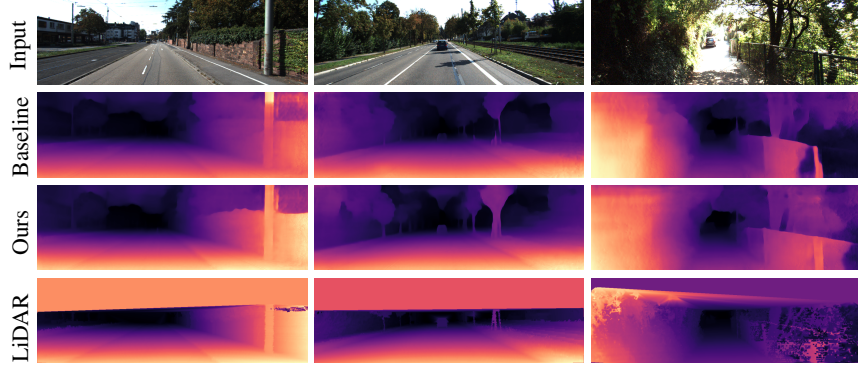
skip connections from the depth encoder to generate a depth prediction and the pose decoder. We also disable updates of all Batch Normalization [37] layers, as we found that updating BatchNorm parameters during inference also causes divergence. We first evaluate each segment separately, then all possible combinations and finally subsections of the most successful network sub-components (see table 1). We conclude that if only weights of the pose and the depth encoders are updated at inference, the depth estimation accuracy improves the most.

**Learning Rate** At training time, the model is trained with Adam [38] with learning rate  $10^{-4}$ , consistently with [9]. Adam however requires warm-up iterations, so at inference time we use vanilla SGD and therefore choosing the correct learning rate to update model parameters is an important consideration, as too low rate results in little to no change, but too high learning rate results in quick divergence. In table 2 we explore a range of reasonable learning rates by taking 10 updates steps and observing the overall accuracy, and we conclude that the learning rate of 0.1 is the best as it results in the most significant improvement in results.

**Number of Optimization Steps** The next natural question is how long we can update network parameters to improve performance while being time-efficient and not causing divergence. Table 3 shows that with learning rate 0.1 we can see that even a low number of steps allows us to still benefit and therefore taking 50 steps results in a good balance between time efficiency and performance improvement.

Steps	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Seconds per Image
Initial	0.115	0.905	4.863	0.193	0.877	0.959	0.981	0.022
5	0.102	0.806	4.566	0.182	0.897	0.962	0.982	0.2
10	0.097	0.788	4.464	0.178	0.905	0.964	0.982	0.544
25	0.091	0.752	4.331	0.174	0.911	0.964	0.982	1.21
50	0.089	0.742	4.263	0.172	0.913	0.964	0.982	1.855
75	0.088	0.735	4.226	0.171	0.914	0.965	0.982	3.551
100	0.086	0.723	4.22	0.17	0.916	0.965	0.982	4.626
150	0.089	0.759	4.219	0.174	0.91	0.962	0.98	6.943
200	0.09	0.786	4.241	0.176	0.908	0.961	0.98	9.355

**Table 3.** Number of optimization steps ablation



**Fig. 3.** More qualitative samples from the KITTI dataset [1]. From top to bottom: monocular input image, the baseline state-of-the-art method [9], our method and the interpolated LiDAR ground truth.

Learning Rate	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
$10^{-4}$	0.115	0.885	4.705	0.19	0.879	0.961	0.982
$10^{-2}$	0.115	0.885	4.705	0.19	0.879	0.961	0.982
1	0.115	0.884	4.7	0.19	0.879	0.961	0.982
10	0.114	0.878	4.677	0.189	0.88	0.962	0.982

**Table 4.** Ablation: direct depth optimization

**Direct Optimization** Instead of optimizing the weights of the network, we can also consider a “naive” approach by optimizing the networks outputs directly. Namely, we take the depth map and pose outputs and optimise them by taking the loss function eq. (1) and using the gradients to update the depth map and the pose estimate directly. As we show in table 4, this naive approach does not improve the baseline. Optimizing only the

Method	Train	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Eigen [16]	D	0.203	1.548	6.307	0.282	0.702	0.890	0.890
Liu [22]	D	0.201	1.584	6.471	0.273	0.680	0.898	0.967
Klodt [39]	D*M	0.166	1.490	5.998	—	0.778	0.919	0.966
AdaDepth [40]	D*	0.167	1.257	5.578	0.237	0.771	0.922	0.971
Kuznetsov [20]	DS	0.113	0.741	4.621	0.189	0.862	0.960	0.986
DVSO [41]	D*S	0.097	0.734	4.442	0.187	0.888	0.958	0.980
SVSM FT [42]	DS	<u>0.094</u>	<u>0.626</u>	4.252	0.177	0.891	0.965	0.984
Guo [43]	DS	0.096	0.641	<u>4.095</u>	<u>0.168</u>	<u>0.892</u>	<u>0.967</u>	<u>0.986</u>
DORN [4]	D	<b>0.072</b>	<b>0.307</b>	<b>2.727</b>	<b>0.120</b>	<b>0.932</b>	<b>0.984</b>	<b>0.994</b>
Zhou [3]†	M	0.183	1.595	6.709	0.270	0.734	0.902	0.959
Yang [44]	M	0.182	1.481	6.501	0.267	0.725	0.906	0.963
Mahjourian [45]	M	0.163	1.240	6.220	0.250	0.762	0.916	0.968
GeoNet [26]†	M	0.149	1.060	5.567	0.226	0.796	0.935	0.975
DDVO [46]	M	0.151	1.257	5.583	0.228	0.810	0.936	0.974
DF-Net [47]	M	0.150	1.124	5.507	0.223	0.806	0.933	0.973
LEGO [48]	M	0.162	1.352	6.276	0.252	—	—	—
Ranjan [49]	M	0.148	1.149	5.464	0.226	0.815	0.935	0.973
EPC++ [35]	M	0.141	1.029	5.350	0.216	0.816	0.941	0.976
Struct2depth ‘(M)’ [24]	M	0.141	1.026	5.291	0.215	0.816	0.945	0.979
Monodepth V2 [9]	M	0.115	0.903	4.863	0.193	0.877	0.959	0.981
Monodepth V2 [9] (1024 × 320)	M	0.115	0.882	4.701	0.190	0.879	<u>0.961</u>	<u>0.982</u>
Patil et al. [28]	M	0.111	0.821	<u>4.65</u>	0.187	0.883	0.961	0.982
GLNet [27]	M	0.135	1.07	5.230	0.210	0.841	0.948	0.980
<i>ours</i>	M	<b>0.089</b>	<b>0.747</b>	4.275	0.173	0.912	0.964	0.982
<i>ours</i> (1024×320)	M	<b>0.089</b>	0.756	<b>4.228</b>	<b>0.170</b>	<b>0.917</b>	<b>0.967</b>	<b>0.983</b>
Garg [19]†	S	0.152	1.226	5.849	0.246	0.784	0.921	0.967
Monodepth R50 [8]†	S	0.133	1.142	5.533	0.230	0.830	0.936	0.970
StrAT [50]	S	0.128	1.019	5.403	0.227	0.827	0.935	0.971
3Net (R50) [51]	S	0.129	0.996	5.281	0.223	0.831	0.939	0.974
3Net (VGG) [51]	S	0.119	1.201	5.888	0.208	0.844	0.941	0.978
SuperDepth+pp [52] (1024×382)	S	0.112	0.875	4.958	0.207	0.852	0.947	<u>0.977</u>
Monodepth V2 [9]	S	0.109	0.873	4.960	0.209	0.864	0.948	0.975
Monodepth V2 [9] (1024 × 320)	S	0.107	0.849	4.764	0.201	0.874	0.953	0.977
D3VO [53]	S	<u>0.099</u>	<u>0.763</u>	<u>4.485</u>	<u>0.185</u>	<u>0.885</u>	<u>0.958</u>	<u>0.979</u>
<i>ours</i>	S	0.076	0.691	4.264	0.182	0.916	0.958	0.976
<i>ours</i> (1024×320)	S	<b>0.075</b>	<b>0.683</b>	<b>4.186</b>	<b>0.179</b>	<b>0.918</b>	<b>0.960</b>	0.978
UnDeepVO [54]	MS	0.183	1.730	6.57	0.268	—	—	—
Zhan FullNYU [55]	D*MS	0.135	1.132	5.585	0.229	0.820	0.933	0.971
EPC++ [35]	MS	0.128	0.935	5.011	0.209	0.831	0.945	0.979
Monodepth V2 [9]	MS	0.106	0.818	4.750	0.196	0.874	0.957	0.979
Monodepth V2 [9] (1024 × 320)	MS	<u>0.106</u>	<u>0.806</u>	<u>4.630</u>	<u>0.193</u>	<u>0.876</u>	<u>0.958</u>	<u>0.980</u>
<i>ours</i>	MS	<b>0.074</b>	<b>0.636</b>	4.082	0.168	0.924	0.966	0.982
<i>ours</i> (1024×320)	MS	0.076	0.638	<b>4.044</b>	<b>0.166</b>	<b>0.925</b>	<b>0.967</b>	<b>0.982</b>

**Legend:** D — Depth supervision, D\* — Auxiliary depth supervision, S — Self-supervised stereo, M — Self-supervised mono, † — Newer results from GitHub, + pp — With post-processing. For **red** metrics, the lower is better; for **blue** metrics, the higher is better. Best results in each category are in **bold**; second-best underlined

**Table 5.** Depth estimation accuracy on the KITTI dataset [1] using the Eigen split [16]

depth map and not the pose prediction performed similarly poor (results not shown for brevity).

#### 4.4 Instance-wise Inference

In the instance-wise strategy, weights are modified individually for each image, without any information from previously observed frames. Resetting weights after adapting to an instance can be beneficial as it removes the possibility of overfitting a sequence very different to generally observed scenes, but may result in lack of temporal consistency. In this experiment, for each image we start by taking the weights obtained by training the baseline method [9] on the training split and we take 50 steps of SGD with learning rate 0.1. Our method consistently beats all the previous methods (see Table 5) by a significant margin. We also present results in higher resolution as [9, 52] show this can result in greater performance, and for our method this in some cases improves accuracy even further.

Using our method, we can also continue to learn on instances of stereo image pairs — in this case we only optimize the depth encoder as pose differences between images are from a known camera baseline. We again consistently outperform all the previous results by a significant margin, and despite being a self-supervised method trained without any ground truth, we come close to DORN [4], the the state-of-the-art *fully-supervised* depth estimation method.

Combining stereo and monocular data in training is also a popular strategy employed in [9, 35, 54]. In this case we warp the stereo pairs using the known baseline and a pose network is used for temporally adjacent pairs, the method therefore utilizes both monocular temporally adjacent frames and stereo pairs in training and instance-wise adaption. Our method also shows significant improvement, demonstrating robustness of our method to improve with various types of instance data (see Table 5 — bottom section).

data split	strategy	Steps	LR	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	time [s]
Eigen [16]	baseline [9]	—	—	0.115	0.903	4.863	0.193	0.877	0.959	0.981	0.022
	Patil et al. [28]	—	—	0.111	0.821	4.65	0.187	0.883	0.961	0.982	—
	Neural RGB-D IMU+GPS [56]	—	—	0.099	0.473	2.829	0.128	0.932	0.980	0.993	0.7
	Neural RGB-D IMU [56]	—	—	0.112	0.449	3.204	0.151	0.893	0.983	0.996	0.7
	Neural RGB-D [56]	—	—	0.120	0.5763	3.516	0.1672	0.878	0.972	0.991	1.5
	ours (instance-wise)	50	0.1	<b>0.089</b>	<b>0.747</b>	<b>4.275</b>	<b>0.173</b>	<b>0.912</b>	<b>0.964</b>	<b>0.982</b>	1.855
	ours (sequence-wise)	5	0.1	0.105	0.896	4.707	0.186	0.895	0.960	0.981	0.2
Odometry	baseline [9]	—	—	0.147	1.090	5.171	0.219	0.805	0.942	0.978	0.022
	ours (instance-wise)	50	0.01	0.116	0.895	<b>4.289</b>	<b>0.181</b>	0.772	0.867	0.893	1.855
	ours (sequence-wise)	5	0.01	<b>0.113</b>	<b>0.887</b>	4.454	0.186	<b>0.874</b>	<b>0.956</b>	<b>0.981</b>	0.2

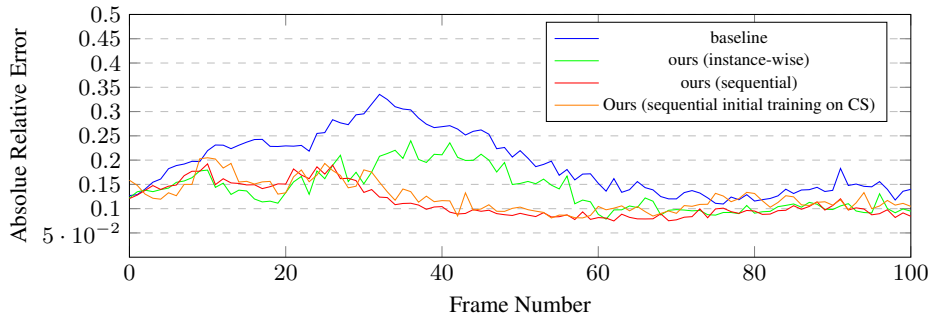
**Table 6.** Qualitative results on the KITTI test set. The Odometry split consist of image sequences of approx. 1000 frames. Neural RGB->D [56] has multiple variants depending on which additional sensors are used, our method has no additional sensors

## 4.5 Sequential Inference

In the sequential strategy, we again start by taking the weights obtained by training the baseline method [9] on the training split for the first frame in the video sequence, but instead of resetting the weights to the pre-trained we allow our weights accumulate information between frames. This not only improves speed of adaption as information from previous frames is highly relevant for the current frame but also improves the consistency of depth maps over time, preventing objects from appearing closer/further away than before. The initialisation of this network uses the same parameters as the above instance method with no modification to training procedure needed, we simply benefit from the similarity of temporally adjacent frames to improve runtime while retaining accuracy as can be seen in table 6. We again evaluate on the Eigen split [16], but because the split contains only short sequences and we want to test robustness of this approach for long sequences, we also evaluate on the KITTI Odometry test split [1], which contains 11 sequences between 491 and 4981 frames in length.

We observe that for both sets, the sequential strategy also consistently improves over the baseline, performing approximately at par with the instance-wise strategy, but *being 10 times faster* (see table 6). The experiment also shows that even for long video sequences, the model still performs well and does not diverge (see fig. 4), despite the fact it had already made thousands of self-supervised updates to the baseline model (e.g. for a video of 4981 frames, the method makes  $4980 \times 5 = 24,900$  updates, so one needs to ask whether the model parameters wouldn’t “drift away” after so many steps).

In [28] they find that using an LSTM on top of networks like Monodepth2 allows them to further refine their depth predictions by exploiting spatio-temporal structure at both train and test time. Our method is not only substantially easier to train to convergence than an LSTM but our method achieves superior results. Neural RGB-D [56] accumulates a depth probability volume over time and uses additional sensors (GPS, IMU), still our method performs better (see table 6) whilst being 2.5-5 times faster.



**Fig. 4.** Absolute depth estimation error for individual frames in the KITTI Odometry 9 sequence



Method	Steps	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
GeoNet [26]	-	0.210	1.723	6.595	0.281	0.681	0.891	0.960
Casser et al. [24]	0	0.205	1.681	6.555	0.275	0.697	0.9	0.961
baseline	0	0.182	2.269	6.454	0.259	0.761	0.919	0.966
GLNet [27] baseline	0	0.206	1.611	6.609	0.281	0.682	0.895	0.959
Casser et al. [24]	20	0.153	1.109	5.557	0.227	0.796	0.934	0.975
Ours	20	0.148	2.111	5.295	0.215	0.842	0.942	0.972
GLNet [27]	50	0.129	1.044	5.361	0.212	0.843	0.938	0.976
Ours	50	<b>0.126</b>	<b>1.038</b>	<b>5.083</b>	<b>0.201</b>	<b>0.855</b>	<b>0.951</b>	<b>0.979</b>

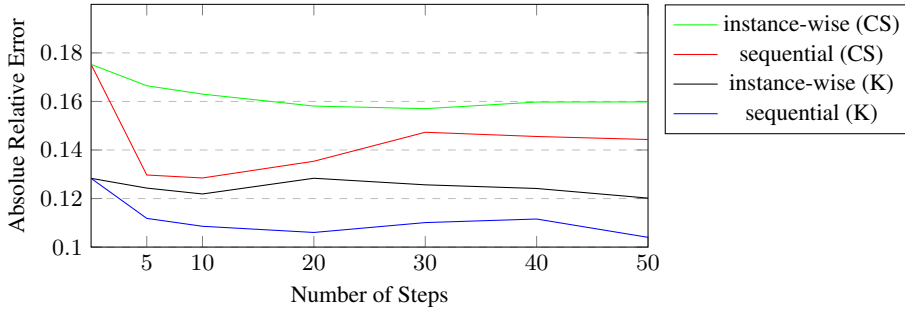
**Table 7.** Qualitative results on the KITTI Eigen split, when *trained on the Cityscapes dataset*. Note that GLNet [27] takes 40 seconds vs 2 seconds (ours) for a single image

#### 4.6 Domain Adaptation

One of the main issues of deep learning models is that test-time examples far from the distribution of the training set often result in poor performance, and test sets from the same dataset can often be misleading as their capture methodology is similar and are typically captured in a similar geographical area. With autonomous driving it is required that a network is capable of adapting to vastly different scenery to bridge the gap from dataset to real life. Even between datasets such as Cityscapes [57] and KITTI [1] generalisation can be poor.

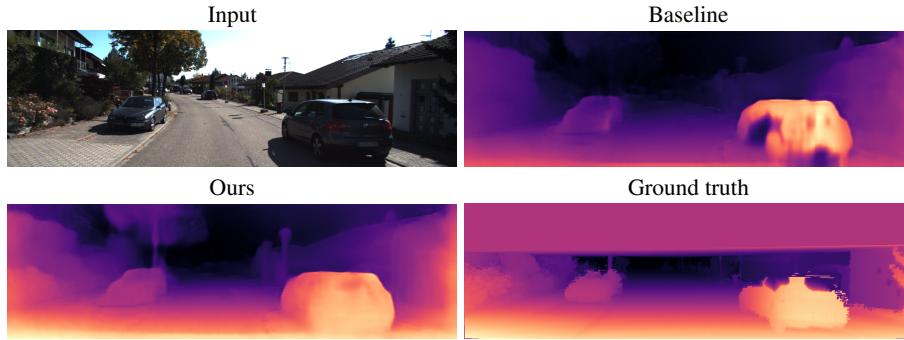
In the first experiment, a baseline model is trained on Cityscapes, but it is evaluated on KITTI, using our test-time instance adaptation. We also compare this to using the baseline without any adaptation and to previous methods [27] (see table 7), and show that our method significantly improves the accuracy of the model when trained on a different dataset, and it outperforms all the prior methods.

In the second experiment, we compare models trained on both Cityscapes and KITTI and evaluate them on KITTI Odometry test set (sequences 9-21), which consists of longer sequences compared to the previous experiment. We then show what is the accuracy of the model, depending on how many optimization steps are taken at test time (see fig. 5



**Fig. 5.** Quantitative results on the KITTI Odometry test set, depending on the number of optimization steps taken (0 steps represents no adaption). CS = Model trained on Cityscapes, K = trained on Kitti Odometry.

and fig. 6). We observe that our Cityscapes trained model with even a mild number of steps can approach the accuracy of networks trained on the target dataset distribution.



**Fig. 6.** A sample from the KITTI Odometry dataset for a model trained on Cityscapes dataset (baseline) and a model with instance adaptation (ours).

## 5 Conclusion

In this paper, we introduced *instance adaptation for self-supervised* depth estimation methods. The instance adaptation allows us to update certain parameters of a trained model dynamically during inference, improving the depth estimate for image instance individually.

Our self-supervised method consistently outperforms all the previous methods by a significant margin for all three common setups (monocular, stereo and monocular+stereo), and comes very close in accuracy when compared to the *fully-supervised* state-of-the-art methods. We also presented the sequential improvement for our method, which is equally accurate but is 10 times faster than the instance-wise approach.

Last but not least, we demonstrated strong domain adaptation potential of our method, by training a model on one dataset (Cityscapes), but testing it on a different dataset (KITTI).

## References

1. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: CVPR. (2012)
2. Poggi, M., Aleotti, F., Tosi, F., Mattoccia, S.: Towards real-time unsupervised monocular depth estimation on cpu. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE (2018) 5848–5854
3. Zhou, T., Brown, M., Snavely, N., Lowe, D.G.: Unsupervised learning of depth and ego-motion from video. In: CVPR. (2017) 1851–1858
4. Fu, H., Gong, M., Wang, C., Batmanghelich, K., Tao, D.: Deep ordinal regression network for monocular depth estimation. In: CVPR. (2018)

5. Yang, G., Hu, P., Ramanan, D.: Inferring distributions over depth from a single image. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (2019)
6. Madhu Babu, V., Das, K., Majumdar, A., Kumar, S.: Undemon: Unsupervised deep network for depth and ego-motion estimation. In: IROS. (2018) 1082–1088
7. Hochberg, C.B., Hochberg, J.E.: Familiar size and the perception of depth. *The Journal of Psychology* **34** (1952) 107–114
8. Godard, C., Mac Aodha, O., Brostow, G.J.: Unsupervised monocular depth estimation with left-right consistency. In: CVPR. (2017)
9. Godard, C., Mac Aodha, O., Firman, M., Brostow, G.J.: Digging into self-supervised monocular depth estimation. In: CVPR. (2019) 3828–3838
10. Mancini, M., Costante, G., Valigi, P., Ciarfuglia, T.A.: Fast robust monocular depth estimation for obstacle detection with fully convolutional networks. In: IROS, IEEE (2016) 4296–4303
11. Bijelic, M., Gruber, T., Ritter, W.: A benchmark for lidar sensors in fog: Is detection breaking down? In: 2018 IEEE Intelligent Vehicles Symposium (IV), IEEE (2018) 760–767
12. : Velodyne hdl64e whitepaper. ([https://velodynelidar.com/lidar/products/white\\_paper/HDL%20white%20paper\\_OCT2007\\_web.pdf](https://velodynelidar.com/lidar/products/white_paper/HDL%20white%20paper_OCT2007_web.pdf)) Accessed: 2019-11-20.
13. : Understanding radar for automotive (ADAS) solutions. (<https://www.pathpartnertech.com/understanding-radar-for-automotive-adas-solutions/>) Accessed: 2019-11-20.
14. Flynn, J., Neulander, I., Philbin, J., Snavely, N.: Deep stereo: Learning to predict new views from the world’s imagery. In: CVPR. (2016) 5515–5524
15. Scharstein, D., Szeliski, R., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV* **47** (2002) 7–42
16. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: ICCV. (2015) 2650–2658
17. Xie, J., Girshick, R., Farhadi, A.: Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In: ECCV, Springer (2016) 842–857
18. Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., Navab, N.: Deeper depth prediction with fully convolutional residual networks. In: 2016 Fourth international conference on 3D vision (3DV), IEEE (2016) 239–248
19. Garg, R., BG, V.K., Carneiro, G., Reid, I.: Unsupervised cnn for single view depth estimation: Geometry to the rescue. In: ECCV, Springer (2016) 740–756
20. Kuznetsov, Y., Stückler, J., Leibe, B.: Semi-supervised deep learning for monocular depth map prediction. In: CVPR. (2017) 2215–2223
21. Wang, P., Shen, X., Lin, Z., Cohen, S., Price, B., Yuille, A.L.: Towards unified depth and semantic prediction from a single image. In: CVPR. (2015)
22. Liu, F., Shen, C., Lin, G., Reid, I.: Learning depth from single monocular images using deep convolutional neural fields. *IEEE TPAMI* **38** (2015) 2024–2039
23. Lee, J.H., Han, M.K., Ko, D.W., Suh, I.H.: From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326* (2019)
24. Casser, V., Pirk, S., Mahjourian, R., Angelova, A.: Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In: AAAI. (2019)
25. Casser, V., Pirk, S., Mahjourian, R., Angelova, A.: Unsupervised monocular depth and ego-motion learning with structure and semantics. In: CVPR Workshops. (2019) 0–0
26. Yin, Z., Shi, J.: Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In: CVPR. (2018) 1983–1992
27. Chen, Y., Schmid, C., Sminchisescu, C.: Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. *ICCV* (2019) 7062–7071

28. Patil, V., Gansbeke, W.V., Dai, D., Gool, L.V.: Don't forget the past: Recurrent depth estimation from monocular video (2020)
29. Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., Madry, A.: Exploring the landscape of spatial robustness. In: ICML. (2019) 1802–1811
30. Azulay, A., Weiss, Y.: Why do deep convolutional networks generalize so poorly to small image transformations? *J. Mach. Learn. Res.* **20** (2019) 184:1–184:25
31. Jaderberg, M., Simonyan, K., Zisserman, A., kavukcuoglu, k.: Spatial transformer networks. In Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R., eds.: *Advances in Neural Information Processing Systems* 28. Curran Associates, Inc. (2015) 2017–2025
32. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. *TIP* (2004)
33. Zhao, H., Gallo, O., Frosio, I., Kautz, J.: Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging* **3** (2016) 47–57
34. Joo, H., Neverova, N., Vedaldi, A.: Exemplar fine-tuning for 3d human pose fitting towards in-the-wild 3d human pose estimation (2020)
35. Luo, C., Yang, Z., Wang, P., Wang, Y., Xu, W., Nevatia, R., Yuille, A.L.: Every pixel counts ++: Joint learning of geometry and motion with 3d holistic understanding. *IEEE TPAMI* (2019)
36. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016) 770–778
37. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F., Blei, D., eds.: *ICML. Volume 37 of Proceedings of Machine Learning Research.*, Lille, France, PMLR (2015) 448–456
38. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. (2015)
39. Klodt, M., Vedaldi, A.: Supervising the new with the old: learning SFM from SFM. In: ECCV. (2018)
40. Nath Kundu, J., Krishna Uppala, P., Pahuja, A., Babu, R.V.: AdaDepth: Unsupervised content congruent adaptation for depth estimation. In: CVPR. (2018)
41. Yang, N., Wang, R., Stückler, J., Cremers, D.: Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In: ECCV. (2018)
42. Luo, Y., Ren, J., Lin, M., Pang, J., Sun, W., Li, H., Lin, L.: Single view stereo matching. In: CVPR. (2018)
43. Guo, X., Li, H., Yi, S., Ren, J., Wang, X.: Learning monocular depth by distilling cross-domain stereo networks. In: ECCV. (2018)
44. Yang, Z., Wang, P., Xu, W., Zhao, L., Nevatia, R.: Unsupervised learning of geometry with edge-aware depth-normal consistency. In: AAAI. (2018)
45. Mahjourian, R., Wicke, M., Angelova, A.: Unsupervised learning of depth and ego-motion from monocular video using 3D geometric constraints. In: CVPR. (2018)
46. Wang, C., Buenaposada, J.M., Zhu, R., Lucey, S.: Learning depth from monocular videos using direct methods. In: CVPR. (2018)
47. Zou, Y., Luo, Z., Huang, J.B.: DF-Net: Unsupervised joint learning of depth and flow using cross-task consistency. In: ECCV. (2018)
48. Yang, Z., Wang, P., Wang, Y., Xu, W., Nevatia, R.: LEGO: Learning edge with geometry all at once by watching videos. In: CVPR. (2018)
49. Ranjan, A., Jampani, V., Kim, K., Sun, D., Wulff, J., Black, M.J.: Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In: CVPR. (2019)
50. Mehta, I., Sakurikar, P., Narayanan, P.: Structured adversarial training for unsupervised monocular depth estimation. In: 3DV. (2018)
51. Poggi, M., Tosi, F., Mattoccia, S.: Learning monocular depth estimation with unsupervised trinocular assumptions. In: 3DV. (2018)

52. Pillai, S., Ambrus, R., Gaidon, A.: Superdepth: Self-supervised, super-resolved monocular depth estimation. In: ICRA. (2019)
53. Yang, N., von Stumberg, L., Wang, R., Cremers, D.: D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In: CVPR. (2020)
54. Li, R., Wang, S., Long, Z., Gu, D.: Undeepvo: Monocular visual odometry through unsupervised deep learning. In: IEEE ICRA. (2018) 7286–7291
55. Zhan, H., Garg, R., Weerasekera, C.S., Li, K., Agarwal, H., Reid, I.: Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In: CVPR. (2018)
56. Liu, C., Gu, J., Kim, K., Narasimhan, S., Kautz, J.: Neural rgb->d sensing: Depth and uncertainty from a video camera. In: CVPR. (2019)
57. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016)

## Chapter 5

# Calibrating self-supervised monocular depth estimation

This paper was presented at the NeurIPS 2020 Workshop on Machine Learning for Autonomous Driving 2020

---

# Calibrating Self-supervised Monocular Depth Estimation

---

**Robert McCraith**  
robert@robots.ox.ac.uk  
University of Oxford

**Lukas Neumann**  
lukas@robots.ox.ac.uk  
University of Oxford

**Andrea Vedaldi**  
vedaldi@robots.ox.ac.uk  
University of Oxford

## Abstract

In the recent years, many methods demonstrated the ability of neural networks to learn depth and pose changes in a sequence of images, using only self-supervision as the training signal. Whilst the networks achieve good performance, the often over-looked detail is that due to the inherent ambiguity of monocular vision they predict depth up to a unknown scaling factor. The scaling factor is then typically obtained from the LiDAR ground truth at test time, which severely limits practical applications of these methods.

In this paper, we show that incorporating prior information about the camera configuration and the environment, we can remove the scale ambiguity and predict depth directly, still using the self-supervised formulation and not relying on any additional sensors.

## 1 Introduction

Depth estimation is an important computer vision problem with applications in robotics, autonomous driving, augmented reality and scene understanding Poggi et al. [2018], Zhou et al. [2017], Fu et al. [2018], Yang et al. [2019], Madhu Babu et al. [2018]. Of particular theoretical and practical interest is estimating depth from a single RGB image, also known as *monocular depth estimation*. When multiple views are available, depth can be inferred from geometric principles by triangulating image correspondences; however, when only a single view is available, triangulation is not possible and the problem is ill posed. Despite this difficulty, reliable and accurate monocular depth estimation is critical for safety in many applications, with autonomous vehicles being a prime example.

Notably, humans are perfectly able to drive a car with just one eye, suggesting that they can infer depth well enough even from a single view. However, doing so requires prior information on the visual appearance and real-world sizes of typical scene elements, which can then be used to estimate the distance of known objects from the camera. In machine vision, this prior can be learned from 2D images labelled with ground-truth 3D information, extracted from a different sensing modality such as a LiDAR. When using additional sensors is impractical, self-supervised learning can be used instead Godard et al. [2017, 2019], Zhou et al. [2017], Mancini et al. [2016]. In *self-supervised learning*, certain relations or consistency of inputs, rather than ground truth labels, are exploited to train the system. In depth estimation, one typically uses the consistency between subsequent video frames or between stereo image pairs Godard et al. [2017], Zhou et al. [2017].

Whilst recent self-supervised monocular depth estimation methods achieve impressive performance, approaching fully-supervised systems, they all share an important practical limitation which limits

their usefulness in real-world applications. Since reconstructing 3D geometry from images has an inherent scale ambiguity Faugeras et al. [2001], and since self-supervised methods only use visual inputs for training Godard et al. [2017, 2019], they do not predict the depth map  $d_I$  directly, but rather a scaled version  $\Phi(I)$  of it relate to the true depth by an unknown scaling factor  $\alpha_I$ , in the sense that  $d_I = \alpha_I \Phi(I)$ . For evaluation, the scaling factor  $\alpha_I$  is not predicted but calculated at *test time from the ground truth*, usually as the ratio between the median of the predicted depth values and the median of the ground-truth depth values. Furthermore, a different scaling factor is computed for each test image individually Godard et al. [2017]. However, in practical applications ground truth 3D data is not available to calibrate the system, especially in production. Thus, the problem is how to *calibrate* self-supervised depth estimation in order to obtain a physically-accurate prediction, without requiring the use of additional sensors. In this paper, we show that, in a driving scenario, this problem can indeed be solved reliably, robustly and efficiently assuming only knowledge of the camera intrinsics and a very limited amount of additional prior information on the geometry of the system. The output of our technique is a properly calibrated depth map, expressed in meters. The method is applicable to any self-supervised training paradigm and does not require any additional 3D ground truth at training or testing time. This is different from previous monocular depth estimation methods which discounting the scale ambiguity at test time, or use additional sensors to remove it Guizilini et al. [2020]. Thus, we make two key contributions in this paper. First, we bring to the attention of the computer vision community the problem of calibrating self-supervised monocular depth estimation systems without resorting to additional sensors such as LiDARs. This is of clear importance if we wish these systems to be of direct practical value. Furthermore, we analyze to what extent the state-of-the-art existing methods depend on the availability of such data. Second, we propose a simple and yet very efficient calibration technique that *does not* make use of any additional sensors, especially of a complex and expensive nature such as a LiDAR. Instead, our method is ‘vision closed’ at training as well as a test time, in the sense that, just like self-supervised monocular depth estimation methods, it only requires images as input. The only additional information required for calibration is the approximate knowledge of a single constant which is trivially obtained from the construction of the system. The rest of the paper is structured as follows. In section 2 we give an overview of the state of the art, in section 3 our method is presented. Experimental validation is given in section 4 and the paper is concluded in section 5.

## 2 Previous Work

**Depth Estimation.** Because of the scale ambiguity inherent to predicting the depth from a single image, monocular depth estimation is an ill-posed problem and other (prior) knowledge has to be incorporated to remove the ambiguity. Scharstein et al. [2002] and more recently Flynn et al. [2016] use classical geometry to extract point-to-point matches between images and use triangulation to estimate depth.

The emergence of deep learning re-formulated the problem as a dense scene segmentation problem, where each image pixel is directly assigned a real value corresponding to the depth. A deep network is then trained to predict depth using supervision either from LiDAR, a RGB-D camera or a stereo pair. Eigen and Fergus [2015] regress depth in multiple scales, refining the depth maps from low to high spatial resolution. Xie et al. [2016] improve network architecture by adding skip connections, so that the network can also benefit from high resolution information. Laina et al. [2016] use de-convolutional segments to refine depth in a coarse-to-fine manner Garg et al. [2016], Kuznetsov et al. [2017], while Garg et al. [2016] use CRFs to improve fine details. Fu et al. [2018] introduced a novel discretized depth representation, which modifies the typical regression task into a classification problem, and using a novel loss function (ordinal regression) they significantly improve accuracy. More recently, Lee et al. [2019] use local planar guidance layers to improve performance. In practice, there are indeed other modalities/sensors that can be used to get depth — such as LiDAR, radar or a stereo camera pair. They all however have their own limitations: LiDARs are quite sensitive to weather Bijelic et al. [2018], the depth maps are sparse which may not be enough for far-away or small objects and they have low refresh rate. Similarly, radars suffer from reflections and interference and they struggle to detect small or slowly-moving objects rad. Stereo is very sensitive to precise calibration and the two cameras can become misaligned over time, greatly reducing the depth map accuracy. Generalizing, there is a clear need for redundancy — a moving autonomous vehicle or a robot in urban environment cannot simply stop working if for example one of the cameras becomes occluded by dirt or if the weather is not perfect, as that would be potentially very unsafe. By having a



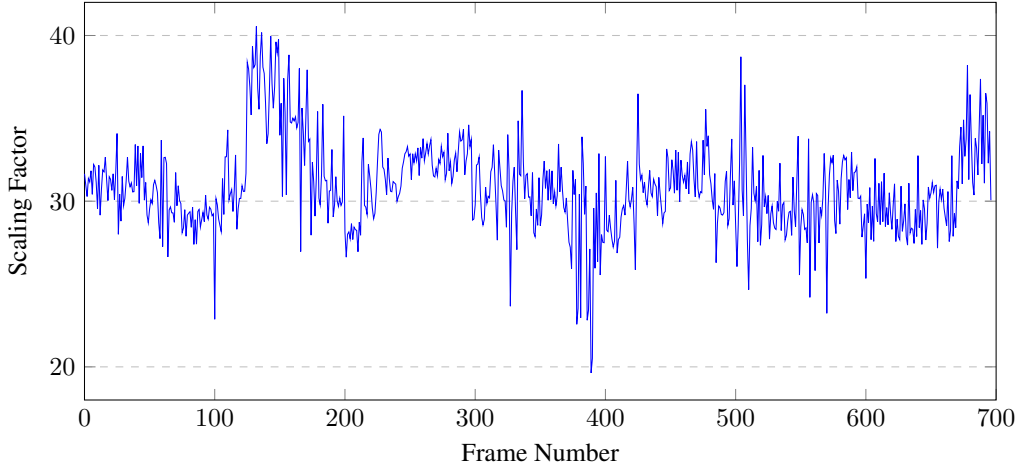


Figure 1: Scaling factor inferred from LiDAR ground truth for every image in the KITTI **test** subset, as used in Godard et al. [2019]

reliable monocular method, this can be used in sensor fusion or as a fallback method, thus improving the overall safety of the system.

**Self-Supervised Methods.** The main struggle for the monocular depth methods is the requirement of vast amounts of training data. The ground truth is typically captured by LiDARs, but this is expensive especially if large variety of driving scenarios and countries has to be covered, and the output is only a sparse point cloud. To alleviate the requirement of having expensive ground truth, recently there has been a surge in interest in unsupervised methods for depth map prediction. Xie et al. [2016] used stereo images in training discrete values for VR and 3D video applications and Garg et al. [2016] extended this approach to continuous values. More recently, Godard et al. [2017] added a left-right depth consistency and Zhou et al. [2017] generalized the approach to monocular sequences at training time, by predicting pose change between two sequential video frames. The sequential nature of the data however introduced some new challenges especially for non-stationary objects, which was addressed in Godard et al. [2019], that incorporates a loss which automatically excludes pixels which have become occluded or which correspond to moving objects. Similarly, Casser et al. [2019a] decompose the image into rigid and non-rigid component, thus reducing the re-projection error. All the above methods Godard et al. [2017, 2019] however share the same weakness which severely limits their applicability — the depth estimate is not calibrated, unlike the supervised methods Fu et al. [2018]. In other words, the depth values of self-supervised methods are not in meters but in some arbitrary unit, which moreover *differs frame by frame* (see fig. 1), and therefore these methods cannot be directly used to reason about the surrounding 3D world.

### 3 Method

In this section, we first describe the self-supervised training paradigm used by state-of-the-art monocular depth estimation methods Godard et al. [2017, 2019]. Then, we discuss how current methods deal with the issue of scale ambiguity by accessing ground truth 3D information at test time. Finally, we detail how basic prior information on a vehicle-mounted can be used to calibrate self-supervised depth map predictions to produce depth map with a correct physical scaling without requiring the acquisition of 3D or other information by means of additional sensors.

#### 3.1 Self-supervised Depth Estimation

Given a pair of subsequent video frames  $I^t$  and  $I^{t+1}$  captured by a moving camera, under mild conditions such as Lambertian reflection, the image  $I^t$  is (approximately) a warp (deformed version) of image  $I^{t+1}$  Hartley and Zisserman [2004]. Moreover, the warp depends only on the *geometry and motion* of the scene, captured by the depth map  $D_t$  and the viewpoint change  $(R^t, T^t)$ . In other words, we can write  $I^t \approx \mathcal{W}(I^{t+1}, D_t, R_t, T_t, K)$ , where  $\mathcal{W}$  is a warp Jaderberg et al. [2015] which

depends only on the depth  $D_t$ , the viewpoint change  $(R_t, T_t)$ , and the camera intrinsics  $K$  (which we assume known and constant).

The equation above provides a constraint that can be used to self-supervise a monocular depth estimation network  $\Phi$  from knowledge of the video frames  $I^t$  and  $I^{t+1}$  alone. In more detail, we task two networks  $\Phi$  and  $\Psi$  to predict respectively the depth  $D^t = \Phi(I^t)$  from the first image and the motion  $(R_t, T_t) = \Psi(I^t, I^{t+1})$  from the pair of images so as to correctly warp  $I^{t+1}$  into  $I^t$ , thus establishing the expected visual consistency (see fig. 2). This is done by minimizing the appearance loss between the original  $I^t$  and the synthesized image  $\hat{I}^t = \mathcal{W}(I^{t+1}, D_t, R_t, T_t, K)$ :

$$\mathcal{L}(I^t, I^{t+1}) = \alpha E_p(I^t, \hat{I}^t) + E_{\text{dis}}(I^t, \hat{I}^t) \quad (1)$$

$$\hat{I}^t = \mathcal{W}(I^{t+1}; D_t, R_t, T_t, K), \quad D_t = \Phi(I^t), \quad (R_t, T_t) = \Psi(I^t, I^{t+1}) \quad (2)$$

The photometric loss term  $E_p$  in eq. (1) is the SSIM loss Wang et al. [2004], Zhao et al. [2016], whilst the  $E_{\text{dis}}$  term enforces smoothness Godard et al. [2017]. The whole network is trained end-to-end using standard back-propagation.

An analysis of the warp operator  $\mathcal{W}$  Jaderberg et al. [2015] shows that the operator is invariant to multiplying the depth and the translation parameters by a constant  $\alpha$ :

$$\mathcal{W}(I; D, R, T, K) = \mathcal{W}(I; \alpha D, R, \alpha T, K) \quad \alpha \in \mathbb{R}. \quad (3)$$

This shows that the network can only learn depth and translation up to an undetermined scaling factor  $\alpha$ ; in particular, there is no reason for the learned scale to corresponds to the true physical scale of the scene. As a matter of fact, the model is not even forced to learn a scaling factor consistently across different pairs of frames  $(I^t, I^{t+1})$ , which we show in empirically is in fact not the case. In particular, fig. 1 shows that the variation in scaling factor for different frames can be up to a factor of two.

### 3.2 Ground Truth Data used to Scale Depth at Test Time

Since the scale of the predicted depth  $\Phi(I^t)$  is arbitrary, its use in downstream tasks that require a physical understanding of the scene (e.g. in robotics) impossible. Equally, all benchmarks for depth estimation Geiger et al. [2012] also require measurements in real units (meters), and therefore the depth map predictions  $\Phi(I^t)$  *cannot* be assessed directly against these benchmarks. Instead, the common approach is to just marginalize out the scale at test time, finding the factor that best matches the predicted and ground-truth depth *for each test image independently* Godard et al. [2017], Casser et al. [2019b], Luo et al. [2018a], Godard et al. [2019]. Since this ground-truth information is obtained via a sensors such as a LiDAR, this is equivalent to calibrating the method against an additional sensor, which is not a realistic setup.

More formally, given an image  $I$ , the network outputs prediction a  $\Phi(I)$  which is transformed to the final depth estimate as  $d_I = \alpha_I \Phi(I)$  where:

$$\alpha_I = \frac{\text{median } d_I^{\text{gt}}}{\text{median } \Phi(I)} \quad (4)$$

where  $d_I^{\text{gt}}$  is the ground-truth depth map, usually created by projecting sparse LiDAR points onto the image plane, projected with the same viewpoint as the input image  $I$ .<sup>1</sup>

<sup>1</sup>Both images are masked such that the scaling factor is only calculated on points where the LiDAR has read data

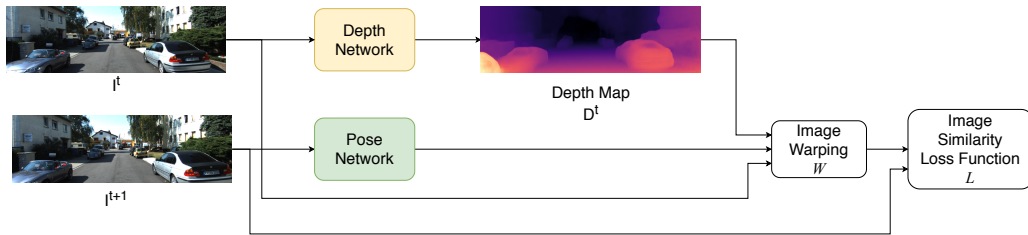


Figure 2: Self-supervised monocular depth estimation pipeline Godard et al. [2017, 2019]

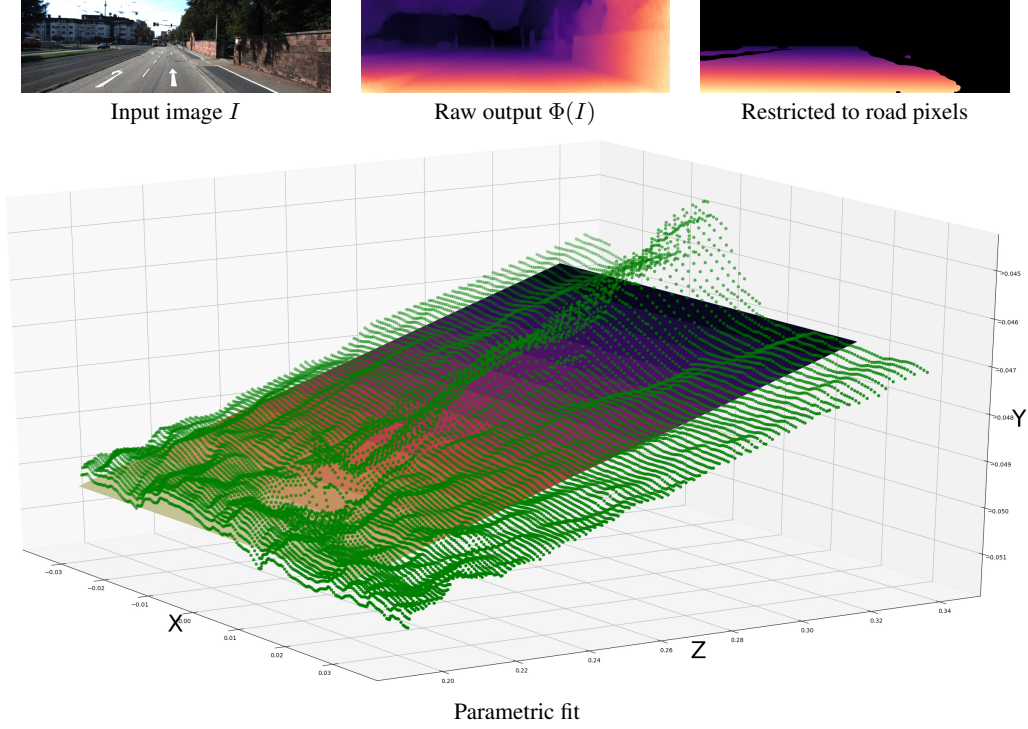


Figure 3: Road model estimation. First we take the uncalibrated depth of the input image, combined with scene segmentation to extract only depth values belonging to the road. After further refinement, we project the points into 3D and fit a plane to them

### 3.3 Road Model Estimation

In order to remove the need for LiDAR ground truth at test time, we exploit prior knowledge of the environment and of the camera setup, especially the camera height. Because cars drive on roads and we know that the camera is at certain height above the road, we can exploit this constraint to calibrate the depth map to real-world values.

In order to do so, we first need to automatically estimate a road model in every test image. In order to account for the fact that many roads are not perfectly flat, more typically they slope up/down or are higher on one side than the other, or that the car tilts during acceleration and deceleration, we estimate the pitch and roll of the road by fitting a plane to the raw depth map. We only use values for pixels that are classified as road by a pre-trained semantic segmentation model Zhou et al. [2018], and whose  $|X|$  and  $Z$  co-ordinate<sup>2</sup> is below a certain threshold (see Section 4.3).

We then fit these points using Least Median of Squares regression Rousseeuw [1984] to get the road plane estimate in the 3D world  $a_1X + a_2Y + a_3Z + c = 0$ . We know that the 3D point on the road right below the camera has the co-ordinate  $[0, -h, 0]$ , where  $h$  is the camera height, and therefore we can infer the following relation for the scaling factor

$$\alpha_I = \frac{c}{h} \quad (5)$$

Compared to eq. (4), the scaling factor eq. (5) now only relies on the visual information, and therefore can be obtained without any LiDAR input.

## 4 Experiments

In this section, we compare our calibration method (Ours) to: (1) the un-calibrated outputs of the network  $D = \Phi(I)$  (Raw); (2) computing a per-frame calibration factor using eq. (4) with

<sup>2</sup>The world co-ordinates  $X, Z$  to filter out pixels above the threshold are obtained using camera intrinsics and assuming the road is perfectly flat, i.e.  $Y = -h$

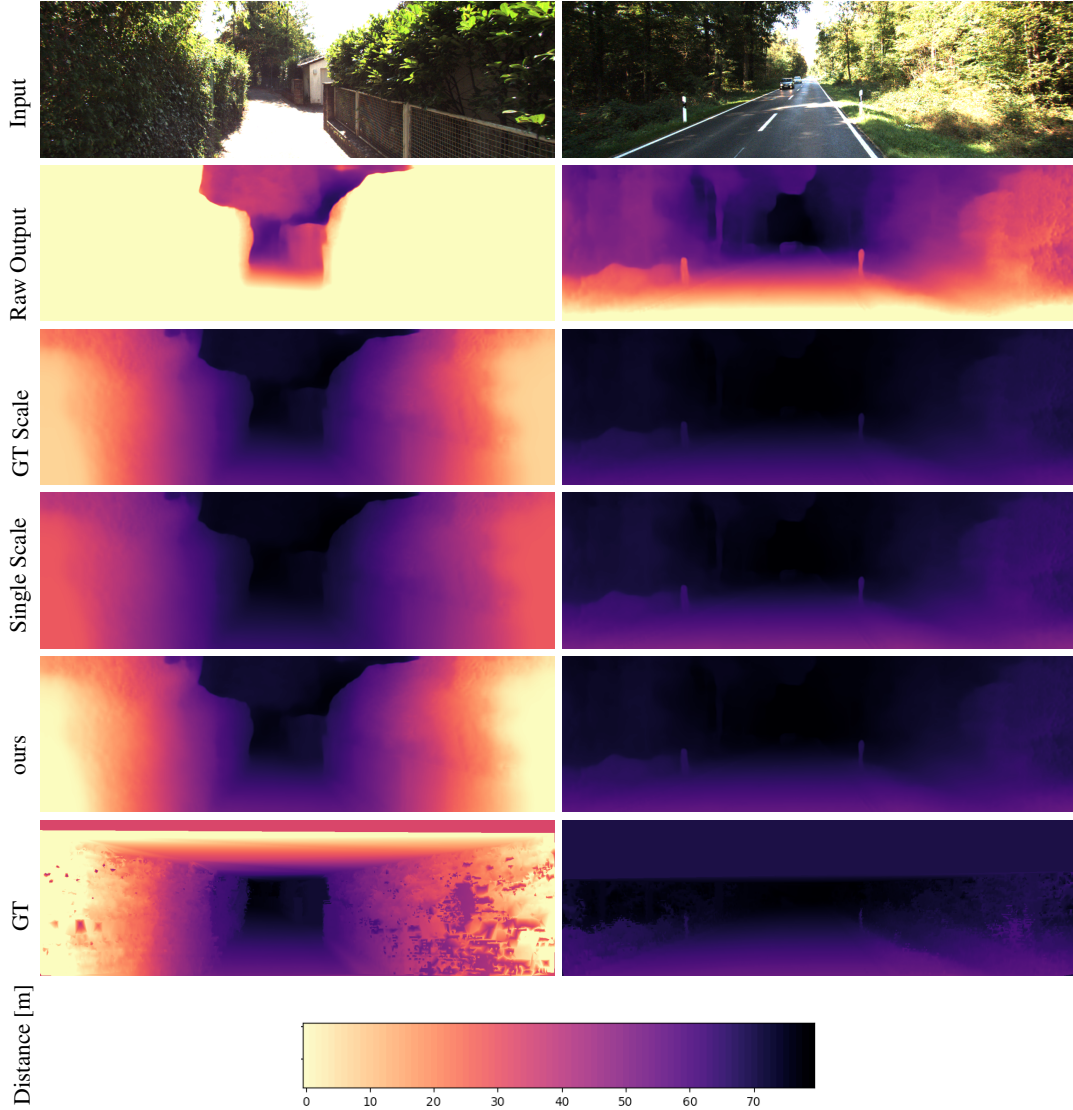


Figure 4: Qualitative depth estimation examples from the KITTI dataset (inverse depth shown). Monodepth2 Godard et al. [2019] output values (Raw Output) are scaled by comparing the output to the ground truth for every test image (GT Scale). Using a single scaling factor from the training set (Single Scale) is significantly worse. Using road model (ours) to estimate the scaling factor achieves significantly better results. All images use the same color coding.

access to the 3D ground-truth (GT Scaling); and (3) the same as (2), but by computing a single scaling factor from either all the training or testing frames (GT Single Scaling). After a qualitative and quantitative comparison with these techniques and state-of-the-art monocular depth estimation networks (both supervised and unsupervised), we ablate our method, showing the importance of the various components, and study sensitivity to its parameters.

**Implementation details.** In all our experiments, we used the Godard et al. [2019] pre-trained model. In line with prior work, we use the Eigen and Fergus [2015] data split of KITTI dataset Geiger et al. [2012].

#### 4.1 Qualitative comparison

We first look at the optimal scaling factor determined via GT Scaling via access to the ground-truth at test time (section 3.2, Godard et al. [2017, 2019], Zhou et al. [2017]). As is observable in Figure 1,

Method	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
(a) GT Single Scaling (training set)	0.125	0.942	5.045	0.208	0.84	0.953	0.979
(b) GT Single Scaling (testing set)	0.126	0.952	4.999	0.204	0.848	0.954	0.98
(c) Fixed road plane	0.132	1.073	5.035	0.203	<b>0.86</b>	<b>0.954</b>	<b>0.977</b>
(d) <i>Ours</i>	<b>0.113</b>	<b>0.916</b>	<b>4.974</b>	<b>0.199</b>	0.857	0.945	0.968

Table 1: Comparison of different depth map scaling methods on the KITTI testing subset.

the factor selected by GT Scaling varies wildly even in a single video sequence. This is illustrated in fig. 4 for two different input images. GT Scaling chooses factor 19.63 for the left image vs GT Single Scaling (median on the training set) of 30.462. This means that, for this image, the network predicts a depth map where objects are 50% farther away than for the median case. The image to the right is the opposite, as GT Scaling determines the best factor to be 40.55, so objects are predicted to be 33% than the median case. Given these differences, it is clear that there is no single scaling factor that results in a good fit for all test frames; hence, below we find it unsurprising that using a single scaling factor over the entire test set (GT Single Scaling) produces inaccurate results overall.

By comparison, our scaling technique predicts scaling factors of 19.63 and 36.4 for the two images respectively, which are close to the output of GT Scaling. Hence, our system produces results significantly closer to the per-frame GT Scaling factors than GT Single Scaling while having *no* access to ground-truth (LiDAR) 3D information at training or test time. This useful for autonomous vehicles that wish to adapt to scenes where it frequently drives rather than examples in a training set as is done in McCraith et al. [2020].

## 4.2 Quantitative comparison

Method	Train	GT@Test	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Eigen and Fergus [2015]	D	<b>X</b>	0.203	1.548	6.307	0.282	0.702	0.890	0.890
Liu et al. [2015]	D	<b>X</b>	0.201	1.584	6.471	0.273	0.680	0.898	0.967
Klodt and Vedaldi [2018]	D*M	<b>X</b>	0.166	1.490	5.998	—	0.778	0.919	0.966
Nath Kundu et al. [2018]	D*	<b>X</b>	0.167	1.257	5.578	0.237	0.771	0.922	0.971
Kuznetsov et al. [2017]	DS	<b>X</b>	0.113	0.741	4.621	0.189	0.862	0.960	0.986
Yang et al. [2018a]	D*S	<b>X</b>	0.097	0.734	4.442	0.187	0.888	0.958	0.980
Luo et al. [2018b]	DS	<b>X</b>	0.094	<u>0.626</u>	4.252	0.177	0.891	0.965	0.984
Guo et al. [2018]	DS	<b>X</b>	0.096	0.641	<u>4.095</u>	0.168	<u>0.892</u>	0.967	<u>0.986</u>
Fu et al. [2018]	D	<b>X</b>	<b>0.072</b>	<b>0.307</b>	<b>2.727</b>	<b>0.120</b>	<b>0.932</b>	<b>0.984</b>	<b>0.994</b>
Zhou et al. [2017] <sup>†</sup>	M	✓	0.183	1.595	6.709	0.270	0.734	0.902	0.959
Yang et al. [2018c]	M	✓	0.182	1.481	6.501	0.267	0.725	0.906	0.963
Mahjourian et al. [2018]	M	✓	0.163	1.240	6.220	0.250	0.762	0.916	0.968
Yin and Shi [2018] <sup>†</sup>	M	✓	0.149	1.060	5.567	0.226	0.796	0.935	0.975
Wang et al. [2018]	M	✓	0.151	1.257	5.583	0.228	0.810	0.936	0.974
Zou et al. [2018]	M	✓	0.150	1.124	5.507	0.223	0.806	0.933	0.973
Yang et al. [2018b]	M	✓	0.162	1.352	6.276	0.252	—	—	—
Ranjan et al. [2019]	M	✓	0.148	1.149	5.464	0.226	0.815	0.935	0.973
Luo et al. [2018a]	M	✓	0.141	1.029	5.350	0.216	0.816	0.941	0.976
Casser et al. [2019b]	M	✓	0.141	1.026	5.291	0.215	0.816	0.945	0.979
Godard et al. [2019]	M	✓	0.115	<b>0.903</b>	<b>4.863</b>	<b>0.193</b>	<b>0.877</b>	<b>0.959</b>	<b>0.981</b>
<i>Ours</i>	M	<b>X</b>	<b>0.113</b>	0.916	4.974	0.199	0.857	0.945	0.968

Table 2: Depth estimation accuracy on the KITTI test set. D — Depth supervision, D\* — Auxiliary depth supervision, M — Self-supervised mono, GT@Test — uses elements of LiDAR ground truth at test time, <sup>†</sup> — Newer results from GitHub, + pp — With post-processing. For **red** metrics, the lower is better; for **blue** metrics, the higher is better. Best results in each category are in **bold**; second-best underlined

First, in table 1 we contrast our method (d) to GT Single Scaling, fixing the scaling factor using respectively the training and testing subset of the data (a) and (b). We note that our approach is substantially better than both (0.113 vs  $\geq 0.125$  AbsRel). This is because, while GT Single Scaling has access to 3D ground-truth, it uses a fixed scaling factor for all frames, and, as shown above, no single scaling factor can work well. Remarkably, our method is comparable to GT Scaling *as well* (the latter corresponds to the penultimate row of table 2), matching it, in particular, in the Abs Rel metric, despite the fact that GT Scaling chooses the best possible scaling factor for each frame



Length	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
6	0.338	2.857	7.47	0.353	0.068	0.132	0.155
10	0.12	0.968	5.013	0.202	0.838	0.933	0.957
15	0.116	0.942	5	0.2	0.853	0.944	0.968
20	0.115	0.932	4.986	0.2	0.856	0.945	0.968
25	0.117	0.956	5.002	0.201	0.856	0.944	0.969
30	0.114	0.926	4.98	0.199	0.856	0.945	0.968
40	0.116	0.933	4.985	0.2	0.856	0.946	0.971
60	0.115	0.928	4.979	0.2	0.857	0.947	0.971
80	0.116	0.941	4.99	0.2	0.857	0.945	0.97

Table 3: Road model distance (length) ablation

Width (m)	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
0.5	0.119	0.944	5.02	0.206	0.838	0.932	0.957
1	0.116	0.926	4.989	0.201	0.845	0.937	0.961
2	0.114	0.918	4.981	0.2	0.856	0.944	0.968
3	0.113	0.916	4.974	0.199	0.857	0.945	0.968
4	0.115	0.936	4.99	0.2	0.856	0.945	0.968
5	0.114	0.923	4.974	0.199	0.858	0.946	0.97
10	0.116	0.933	4.986	0.2	0.855	0.946	0.969
15	0.116	0.933	4.987	0.2	0.855	0.946	0.969

Table 4: Road model width ablation. Points are considered to create the model if  $|X| < \text{Width}$  and the distance is below 30 meters.

individually against the ground-truth. From the same table, we see that this is obtained against a model, Monodepth V2, which is state-of-the-art, resulting for the first time in excellent *calibrated* self-supervised monocular depth estimation from *vision alone*.

### 4.3 Ablation and tuning

**Road model.** Recall that our method is based on estimating the full 3 DoF of the ground plane. First, we test whether this is necessary. In order to do so, we assume instead that the plane is exactly horizontal and at the fixed canonical height below the camera. Then, we use the fixed plane to generate a pseudo-LiDAR map for the road pixels and use GT Scaling against those pseudo-ground-truth values (instead of the actual GT value) in order to determine the scaling factor for each frame. A similar fixed pseudo-LiDAR plane was also used, for example, in Choe et al. [2019] in order to perform 3D object detection. The result of this is shown in table 1 row (c): the fact that this simple fixed-plane model ignores the tendency of real roads to have inclines and declines as well as the cameras ability to have non-negligible pitch and roll during regular car motion which greatly effects it’s depth prediction in the far range.

**Tuning.** Next, we assess the sensitivity of our methods to various parameters and determine their optimal values. In section 3.3 we first take to varying the maximum distance of points on the road we use for the plane fitting. Similar to Man et al. [2019] we find that using points predicted to be under 30 meters from our camera works best for fitting our ground plane as seen in table 4. In a similar fashion we explore the maximum left-right distance from which we consider points from our fit. Typical roads are between 2.75 and 3.75 meters wide so it is within reason that only points within 3m left of right of the car work best with a gradual drop off in performance above this and an insufficient number of points below. Note that the method is not overly sensitive to a specific parameter setting, likely due to the use of robust estimator.

## 5 Conclusion

In this paper, we highlighted the limitation of self-supervised depth estimation methods and their reliance on LiDAR data at test time. We additionally showed how to overcome this issue by incorporating prior information about camera configuration and the environment, and we achieved comparable performance to the state of the art through vision only, without relying on any additional sensors.

## References

- Understanding radar for automotive (ADAS) solutions. <https://www.pathpartnertech.com/understanding-radar-for-automotive-adas-solutions/>. Accessed: 2019-11-20.
- Mario Bijelic, Tobias Gruber, and Werner Ritter. A benchmark for lidar sensors in fog: Is detection breaking down? In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 760–767. IEEE, 2018.
- Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Unsupervised monocular depth and ego-motion learning with structure and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019a.
- Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *AAAI*, 2019b.
- Jaesung Choe, Kyungdon Joo, François Rameau, Gyu Min Shim, and In So Kweon. Segment2regress: Monocular 3d vehicle localization in two stages. In *Robotics: Science and Systems*, 2019.
- David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, pages 2650–2658, 2015.
- Olivier D. Faugeras, Quang-Tuan Luong, and Théodore Papadopoulos. *The geometry of multiple images - the laws that govern the formation of multiple images of a scene and some of their applications*. MIT Press, 2001.
- John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world’s imagery. In *ICCV*, pages 5515–5524, 2016.
- Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, June 2018.
- Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.
- Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, 2012.
- Clément Godard, Oisín Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *ICCV*, pages 270–279, 2017.
- Clément Godard, Oisín Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3828–3838, 2019.
- Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- Xiaoyang Guo, Hongsheng Li, Shuai Yi, Jimmy Ren, and Xiaogang Wang. Learning monocular depth by distilling cross-domain stereo networks. In *ECCV*, 2018.
- R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
- Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.
- Maria Klodt and Andrea Vedaldi. Supervising the new with the old: learning SFM from SFM. In *ECCV*, 2018.
- Yevhen Kuznetsov, Jorg Stuckler, and Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. In *ICCV*, pages 6647–6655, 2017.

- Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, pages 239–248. IEEE, 2016.
- Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019.
- Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE TPAMI*, 38(10):2024–2039, 2015.
- Chenxu Luo, Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, Ram Nevatia, and Alan Yuille. Every pixel counts+: Joint learning of geometry and motion with 3D holistic understanding. *arXiv*, 2018a.
- Yue Luo, Jimmy Ren, Mude Lin, Jiahao Pang, Wenxiu Sun, Hongsheng Li, and Liang Lin. Single view stereo matching. In *CVPR*, 2018b.
- V. Madhu Babu, K. Das, A. Majumdar, and S. Kumar. Undemon: Unsupervised deep network for depth and ego-motion estimation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1082–1088, Oct 2018. doi: 10.1109/IROS.2018.8593864.
- Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3D geometric constraints. In *CVPR*, 2018.
- Yunze Man, Xinshuo Weng, Xi Li, and Kris Kitani. Groundnet: Monocular ground plane normal estimation with geometric consistency. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM ’19, page 2170–2178, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368896. doi: 10.1145/3343031.3351068. URL <https://doi.org/10.1145/3343031.3351068>.
- Michele Mancini, Gabriele Costante, Paolo Valigi, and Thomas A Ciarfuglia. Fast robust monocular depth estimation for obstacle detection with fully convolutional networks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4296–4303. IEEE, 2016.
- Robert McCraith, Lukas Neumann, Andrew Zisserman, and Andrea Vedaldi. Monocular depth estimation with self-supervised instance adaptation, 2020.
- Jogendra Nath Kundu, Phani Krishna Uppala, Anuj Pahuja, and R. Venkatesh Babu. AdaDepth: Unsupervised content congruent adaptation for depth estimation. In *CVPR*, 2018.
- Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. Towards real-time unsupervised monocular depth estimation on cpu. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5848–5854. IEEE, 2018.
- Anurag Ranjan, Varun Jampani, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *CVPR*, 2019.
- Peter J Rousseeuw. Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880, 1984.
- Daniel Scharstein, Richard Szeliski, and Rick Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, May 2002.
- Chaoyang Wang, Jose Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learning depth from monocular videos using direct methods. In *CVPR*, 2018.
- Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004.
- Junyuan Xie, Ross Girshick, and Ali Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *European Conference on Computer Vision*, pages 842–857. Springer, 2016.



- Gengshan Yang, Peiyun Hu, and Deva Ramanan. Inferring distributions over depth from a single image. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.
- Nan Yang, Rui Wang, Jörg Stückler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *ECCV*, 2018a.
- Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, and Ram Nevatia. LEGO: Learning edge with geometry all at once by watching videos. In *CVPR*, 2018b.
- Zhenheng Yang, Peng Wang, Wei Xu, Liang Zhao, and Ramakant Nevatia. Unsupervised learning of geometry with edge-aware depth-normal consistency. In *AAAI*, 2018c.
- Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1983–1992, 2018.
- Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging*, 3(1):47–57, 2016.
- Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal on Computer Vision*, 2018.
- Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1858, 2017.
- Yuliang Zou, Zelun Luo, and Jia-Bin Huang. DF-Net: Unsupervised joint learning of depth and flow using cross-task consistency. In *ECCV*, 2018.

# Part III

## 3D Object Detection

## Chapter 6

# Lifting 2D Object Locations to 3D by Discounting LiDAR Outliers across Objects and Views

This work has been presented at ICRA 2022.

# Lifting 2D Object Locations to 3D by Discounting LiDAR Outliers across Objects and Views

Robert McCraith<sup>1</sup>, Eldar Insafutdinov<sup>2</sup>, Lukas Neumann<sup>3</sup>, Andrea Vedaldi<sup>4</sup>

**Abstract**—We present a system for automatic converting of 2D mask object predictions and raw LiDAR point clouds into full 3D bounding boxes of objects. Because the LiDAR point clouds are partial, directly fitting bounding boxes to the point clouds is meaningless. Instead, we suggest that obtaining good results requires sharing information between *all* objects in the dataset jointly, over multiple frames. We then make three improvements to the baseline. First, we address ambiguities in predicting the object rotations via direct optimization in this space while still backpropagating rotation prediction through the model. Second, we explicitly model outliers and task the network with learning their typical patterns, thus better discounting them. Third, we enforce temporal consistency when video data is available. With these contributions, our method significantly outperforms previous work despite the fact that those methods use significantly more complex pipelines, 3D models and additional human-annotated external sources of prior information.

## I. INTRODUCTION

Robotics applications often require to recover the 3D shape and location of objects in world coordinates. This explains the proliferation of datasets such as KITTI, nuScenes and SUN RGB-D [7], [2], [34] which allow to train models that can classify, detect and reconstruct objects in 3D from sensors such as cameras and LiDARs. However, creating such datasets is very expensive. For example, [32], [39] report that manually annotating a single object with a 3D bounding box requires approximately 100 seconds. While this cost has since been reduced [19], it remains a significant bottleneck in data collection.

In some cases, cross-modal learning can substitute manual annotations. An example is monocular depth prediction, where supervision from a LiDAR sensor is generally sufficient [6]. Unfortunately, this does not extend to tasks such as object detection. For instance, a dataset such as KITTI provides only 7481 video frames annotated with objects due to the cost of manual annotation.

In this work, we thus consider the problem of detecting objects in 3D, thus also automatically annotating them in 3D, but using only standard 2D object detector trained on a generic dataset such as MS-COCO [15], which is disjoint from the task-specific dataset (in our case KITTI for 3D car detection). We assume to have as input a collection of video frames, the corresponding LiDAR readings from the viewpoint of a moving vehicle, and the ego-motion of the vehicle. We also assume to have a pre-trained 2D detector and

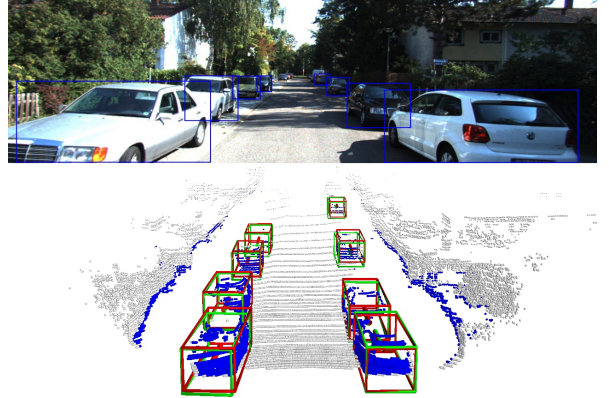


Fig. 1: Our Method (green) vs KITTI label (red), no Ground Truth is used to train our model.

segmenter such as Mask R-CNN for the objects of interest (e.g., cars). With this information, we wish to train a model which takes in raw LiDAR point cloud and outputs full 3D bounding box annotation for the detected objects without incurring any further manual annotation cost.

There are three main challenges. First, due to self and mutual occlusions, the LiDAR point clouds only cover part of the objects in a context dependent manner. Second, the LiDAR readings are noisy, for example sometimes seeing through glass surfaces (windows) and sometimes not. Third, the available 2D segmentations may not be perfectly semantically aligned with the target class (e.g., ‘vehicles’ vs ‘cars’), are affected by a domain shift, and may not be perfectly geometrically with the LiDAR data, resulting in a large number of outlier 3D points arising from background objects.

We propose an approach based on the following key ideas. First, because the LiDAR point cloud can only cover the object partially, it is impossible to estimate the full 3D extent of the object from a single observation of it. Instead, we share information between all predicted 3D boxes in the dataset by *learning a 3D bounding box predictor from all the available data*. We further aid the process by injecting weak prior information in the form of a single fixed 3D mesh template of the object (an ‘average car’), but avoid sophisticated 3D priors employed in prior works [44], [29].

We then introduce three improvements to the ‘obvious’ baseline implementation of this idea. First, we show that a key challenge in obtaining good 3D bounding boxes is to estimate correctly the yaw (rotation) of the object. This is particularly challenging for partial point clouds as several ambiguous fits (generally up to rotations of 90 degrees) often exist. Prior work has addressed this problem by using pre-

<sup>1,2,4</sup> Authors are with Visual Geometry Group, Dept. of Engineering Science, University of Oxford, <sup>3</sup> Lukas is with Faculty of Electrical Engineering, Czech Technical University in Prague. {robert, eldar, lukas, vedaldi}@robots.ox.ac.uk

trained yaw predictors, requiring manual annotation. Here, we learn the predictor automatically from the available data only. To this end, we show that mere local optimization via gradient descent works poorly; instead, we propose to systematically explore a full range of possible rotations for each prediction, backpropagating the best choice every time. We show that this selection process is very effective at escaping local optima and results in excellent, and automated, yaw prediction.

Second, we note that the LiDAR data contains significant outliers. We thus propose to automatically learn the *pattern* of such outliers by predicting a confidence score for each 3D point, treated as a Gaussian observation. These confidences are self-calibrated using a neural network similar to the ones used for point cloud segmentation, configured to model the aleatoric uncertainty of the predictor.

Third, we note that we generally have at our disposal *video data*, which contains significant more information than instantaneous observations. We leverage this information by enforcing a simple form of temporal consistency across several frames.

As a result of these contributions we obtain a very effective system for automatically labelling 3D objects. Our system is shown empirically to outperform relevant prior work by a large margin, all the while being simpler, because it uses less sources of supervision and because it does *not* use sophisticated prior models of the 3D objects nor a large number of 3D models as priors [44], [29]

## II. RELATED WORK

**a) Supervised:** 3D object detection methods assume availability of either monocular RGB images, LiDAR point clouds or both. Here we focus on supervised methods using 3D point cloud inputs. [37], [4] discretise point clouds onto a sparse 3D feature grid and apply convolutions while excluding empty cells. [3] project the point cloud onto the frontal and the top views, apply 2D convolutions thereafter and generate 3D proposals with an RPN [30]. [46] convert the input point cloud into a voxel feature grid, apply a PointNet [28] to each cell and subsequently process it with a 3D fully convolutional network with an RPN head which generates object detections. Frustum PointNets [27] is a two step detector which first detects 2D bounding boxes using these to determine LiDAR points of interest which are filtered further by a segmentation network. The remaining points are then used to infer the 3D box parameters with the centre prediction being simplified by some intermediate transformations in point cloud origins. We are using Frustum PointNets as a backbone for our method.

**b) Weakly Supervised:** Owing to the complexity of acquiring a large scale annotated dataset for 3D object detection many works recently have attempted to solve this problem with less supervision. [19] the required supervision is reduced from the typical  $(X, Y, Z)$  centre,  $yaw$ ,  $(x_1, y_1, x_2, y_2)$  2D box and  $(l, w, h)$  3D size they instead annotate 500 frames with centre  $(X, Z)$  in the Birds Eye View and finely annotating a 534 car subset of these frames to achieve accuracy similar to models trained on the entire Kitti training set. This however can result in examples in the larger

training set or validation set which are outside the distribution of cars seen in the smaller subset, are also susceptible to the problems mentioned in [5] and the weakly annotated centres can have a large difference to the Kitti annotations. In [44] trains DeepSDF[22] on models from a synthetic dataset which are then rendered into the image and iteratively refined to produce a prediction that best fits the predictions of Mask R-CNN outputs. This however takes 6 seconds to refine predictions on each input example and ground truth 2D boxes are used to select cars used to train on. In [29] 3D anchors densely placed across the range of annotations are projected into the image with object proposed by looking at the density of points within or nearby the anchors in 3D and the 3D pose and 2D detections are supervised by a CNN trained on Beyond PASCAL[41]. In [13] instance segmentation is used to place a mesh in the location of detected cars which is refined using supervised depth estimation.

**c) Viewpoint estimation:** Estimating 3D camera orientation is an actively researched topic [25], [12], [36], [33], [20], [26], [14]. Central to this problem is the choice of an appropriate representation for rotations. Directly representing rotations as angles  $\theta \in [0, 2\pi)$  suffers from discontinuity at  $2\pi$ . One way to mitigate this is to use the trigonometric representation  $(\cos \theta, \sin \theta)$  [24], [17], [1]. Another approach is to discretise the angle into  $n$  quantised bins and treat viewpoint prediction as a classification problem [36], [33], [20]. Quaternions are another popular representation of rotations used with neural networks [12], [43]. Learning camera pose without direct supervision, for example, when fitting a 3D model to 2D observations, may suffer from ambiguities due to the object symmetries [31]. Practically, this means that the network can get stuck in a local optima of  $SO(3)$  space and not be able to recover the correct orientation. Several recent works [35], [10], [8] overcome this by maintaining several diverse candidates for the estimated camera rotation and selecting the one that yields the best reconstruction loss. Our approach discussed in sec. III-C is most related to these methods.

## III. METHOD

Our goal is to estimate the 3D bounding box of objects given as input videos with 2D mask predictions and LiDAR point clouds. We discuss first how this problem can be approached by direct fitting and then develop a much better learning-based solution.

### A. Shape model fitting

We first describe how a 3D bounding box can be fitted to the available data in a direct manner. To this end, let  $I \in \mathbb{R}^{3 \times H \times W}$  be a RGB image obtained from the camera sensor and let  $L \subset \mathbb{R}^3$  be a corresponding finite collection of 3D points  $\mathbf{X} \in L$  extracted from the LiDAR sensor. Furthermore, let  $m \in \{0, 1\}^{H \times W}$  be the 2D mask of the object obtained from a system such as Mask R-CNN [9] from image  $I$ . Our goal is to convert the 2D mask  $m$  into a corresponding 3D bounding box  $B$ .

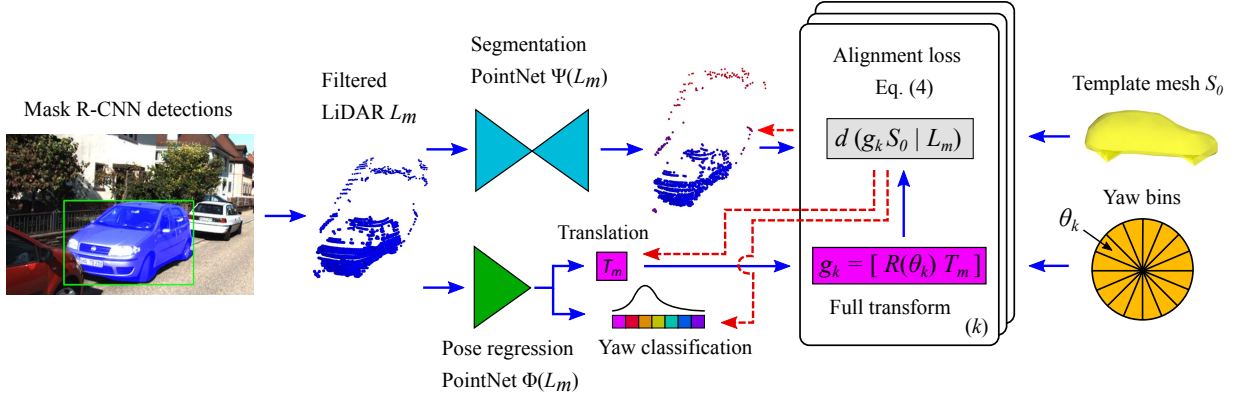


Fig. 2: Network architecture. Dashed arrows in red show the flow of the gradients during the backward pass. Alignment loss is evaluated for each yaw bin  $\theta_k$  and the optimal yaw is used to supervise the yaw classification network  $\Phi_r$  (see sec. III-C).

To do so, assume that the LiDAR points are expressed in the reference frame of the camera and that the camera calibration function  $k: \mathbb{R}^2 \rightarrow \Omega = \{1, \dots, H\} \times \{1, \dots, W\}$  is known. The calibration function is defined such that the 3D point  $\mathbf{X} = (X, Y, Z)$  projects onto the image pixel  $u = k(\pi(\mathbf{X}))$  where  $\pi(X, Y, Z) = (X/Z, Y/Z)$  is the perspective projection. In particular, the subset of LiDAR points  $L_m \subset L$  that project onto the 2D mask  $m$  is given by:  $L_m = L \cap (k \circ \pi)^*(m)$  where  $*$  denotes the pre-image of a function. In practice, this is a crude filtering step, because the masks are imprecise and not perfectly aligned to the LiDAR and because LiDAR may sometimes see ‘through’ the object, for instance in correspondence of glass surfaces (see Fig. 3).

In order to fit a 3D bounding box  $B$  to  $L_m$ , we use a weak prior on the 3D shape of the object. Specifically, we assume that a 3D template surface  $S_0 \subset \mathbb{R}^3$  is available, for example as simplicial (triangulated) mesh. We fit the 3D surface to the LiDAR points by considering a rigid motion  $g \in SE(3)$  which, applied to  $S_0$ , results in the posed mesh  $S = gS_0 = \{g\mathbf{X} : \mathbf{X} \in S_0\}$ . We then define a distance between the mesh and the 3D LiDAR points as follows:

$$d(S|L_m) = \frac{1}{|L_m|} \sum_{\mathbf{X} \in L_m} \min_{\mathbf{X}' \in S} \|\mathbf{X}' - \mathbf{X}\|^2. \quad (1)$$

This quantity is similar to a Chamfer distance, but it only considers half of it: this is because most of the 3D points that



Fig. 3: For each pair, left: RGB input image with Mask R-CNN predicted box and highlighted pixels inside the mask. Right: LiDAR points in blue represent those inside the 2D mask, green those outside. Note that, while the image mask removes many outliers, many remain at the object boundaries and transparent surfaces.

belong to the template object are *not* be visible in a given view (in particular, at least half are self-occluded), so not all points in the template mesh have a corresponding LiDAR point.

Given a 2D object mask  $m$  and its corresponding LiDAR points  $L_m$ , we can find the pose  $g$  of the object by minimizing  $d(gS_0|L_m)$  with respect to  $g \in SE(3)$ . Then the bounding box of the object  $m$  is given by  $gB_0$  where  $B_0 \subset \mathbb{R}^3$  is the 3D bounding box that tightly encloses the template  $S_0$ .

In accordance with prior work [7], [29], [19], we can in practice carry out the minimization not over the of full space  $SE(3)$ , but only on 4-DoF transformation  $g = [R_\theta, \mathbf{T}]$  where the rotation  $R_\theta$  is restricted to the yaw  $\theta$  (rotation perpendicular to the ground plane). Even so, direct minimization of (1) is in practice prone to failure because individual partial LiDAR point clouds do not contain sufficient information and fitting results are thus ambiguous (we do not report results in this setting as they are extremely poor).

Our solution to the ambiguity of fitting (1) is to *share information* across all object instances in the dataset. We do this by training a deep neural network  $\Phi: \text{Fin}(\mathbb{R}^3) \rightarrow SE(3)$  mapping the LiDAR points  $L_m$  to the corresponding object pose  $g = \Phi(L_m)$  directly. The network  $\Phi$  can be trained in a self-supervised manner by minimizing (1) averaged over the entire dataset  $\mathcal{D}$  as

$$\mathcal{L}(\Phi|\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(L_m, m) \in \mathcal{D}} d(g_m S_0 | L_m) \quad (2)$$

where  $g_m = \Phi(L_m)$ .

### B. Modelling and discounting outliers

A major drawback of (2) is that LiDAR points tend to be noisy, especially because the boundaries of the region  $m$  may not correspond to the object exactly or the LiDAR may be affected by a reflection or ‘see through’ a glass surface. Such points might disproportionally skew the loss term, forcing the estimated object position closer to these outliers. In order to help the model discriminate between inliers and outliers, we let the network predict an estimate of whether a given measurement is likely to belong to the object or not.



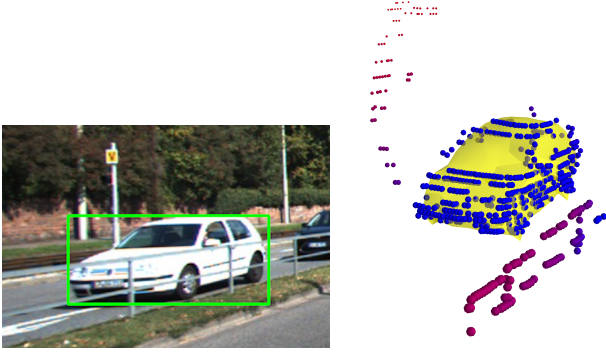


Fig. 4: Even after removing LiDAR points which do not project into the 2D instance mask we still have a large number of outliers, mainly caused by partial occlusions and points at the boundary where 2D masks struggle. The  $\sigma^2(\mathbf{X})$  value predicts the relevance of a point to the cars location with blue points having low values with a gradient to red for higher values which suggests the point is an outlier.

We therefore propose a second network  $\sigma(\mathbf{X}) = \Psi_{\mathbf{X}}(L_m)$  that assigns to each LiDAR point  $\mathbf{X} \in L_m$  a variance  $\sigma$  and jointly optimize  $\Phi$  and  $\Psi$  by minimizing [21], [11]:

$$\bar{d}(S|L_m) = \frac{1}{|L_m|} \sum_{\mathbf{X} \in L_m} \min_{\mathbf{X}' \in \tilde{S}} \frac{\|\mathbf{X}' - \mathbf{X}\|^2}{\sigma^2(\mathbf{X})} + \log \sigma^2(\mathbf{X}) \quad (3)$$

Note that the network  $\Psi$  has to make a judgement call for every point  $\mathbf{X}$  on whether it is likely to be an outlier or not *without* having access to the loss. A perfect prediction (i.e., one that minimizes the loss) would set  $\sigma(\mathbf{X}) = \|\mathbf{X}' - \mathbf{X}\|$  to be the same as the fitting error. The desirable side effect is that, in this manner, outliers are discounted by a large  $\sigma(\mathbf{X})$  when it comes to estimating the pose  $g_m$  of the object.

### C. Direct optimization for the yaw

Finally, we note that fitting the rotation of the object can be ambiguous, especially if only a small fragment of the object is visible in the RGB/LiDAR data. Specifically, the distance  $\bar{d}([R_{\theta_m} \mathbf{T}_m]S_0|L_m)$ , which is usually well behaved for the translation component  $\mathbf{T}_m$ , has instead a number of ‘deep’ local minima, which we found is mainly caused by the inherent ambiguities of fitting the rotation  $\theta_m$  (yaw) parameter. Each minimum corresponds to a  $90^\circ$  rotation as in many example only a single side of the vehicle is visible. In practice, as we show, the yaw network  $\Psi$  *can* learn to disambiguate the prediction, but it usually fails to converge to such a desirable solution without changes in the formulation.

In order to solve this issue, we propose to modify the formulation to incorporate *direct optimisation* over the yaw. In other words, every time the loss is evaluated, we assess a number of possible rotations  $R$ , as follows:

$$R_m^* = \operatorname{argmin}_{R \in \mathcal{R}} \bar{d}([R \mathbf{T}_m]S_0|L_m) \quad (4)$$

$$\bar{r} = \|R_m - R_m^*\| \quad (5)$$

$$\mathcal{L}'(\Phi, \Psi|\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(L_m, m) \in \mathcal{D}} \bar{d}([R_m^* \mathbf{T}_m]S_0|L_m) + \bar{r} \quad (6)$$

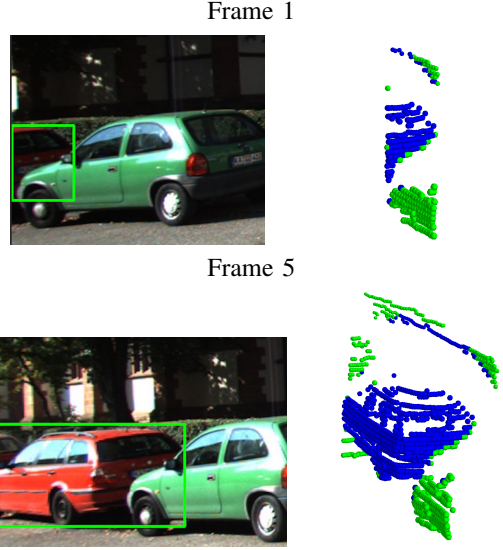


Fig. 5: Two frames in a sequence of five, with a heavily occluded and truncated car in the first frame and a much better view of the same in the last. Multi-frame consistency allows the method to use clearer frames to interpret more difficult ones, which is particularly helpful in discovering the outliers.

The loss is the same as before for the translation, but, given the predicted translation, it always explores all possible rotations  $R$ , picking the best one  $R_m^*$ .

Note that this does not mean that the network  $\Phi$  is not tasked with predicting a rotation anymore; on the contrary, the network is encouraged to output the optimal  $R_m^*$  via minimization of the term  $\|R_m - R_m^*\|$ . This has the obvious benefit of not incurring the search at test time, and therefore the final network running time is not affected by this process.

In practice, as only the yaw angle is predicted, we implement this loss by quantizing the interval  $[0, 2\pi)$  in 64 distinct values (bins), as in our experiments we found this number of bins sufficient (see results in table II). In this manner, the rotation head of the network  $\Phi$  can be interpreted as a softmax distribution  $\Phi_r(L_m)$  and the norm  $\|\cdot\|$  is replaced by the cross-entropy loss.

### D. Multi-frame consistency

Whilst in our approach we do not have the actual 3D pose of the object available as a training signal, the 3D pose of the object across multiple frames must be consistent with the observer ego-motion. This is true for vehicles that are not moving (parked cars) and approximately true for other vehicles; in particular, the *yaw* of the objects, once ego-motion has been compensated for, is roughly constant.

Specifically, consider an arbitrary model point  $\mathbf{X}_0 \in S_0$ . Observed in a LiDAR frame  $i$ , this point is estimated to be at location  $\mathbf{X}_0^i = g^i \mathbf{X}_0$ , where  $g^i$  is the network prediction for frame  $i$ . Likewise, let  $\mathbf{X}_0^j = g^j \mathbf{X}_0$  be the point position at frame  $j$ . If the object is at rest, the two positions are related by  $\mathbf{X}_0^i = g_{i \leftarrow j} \mathbf{X}_0^j$  where  $g_{i \leftarrow j}$  is the ego-motion between frames  $i$  and  $j$  of the vehicle where the sensors are mounted, which

we assume to be known. We can thus define the consistency loss  $\mathcal{L}^{\mathbf{X}_0}$  for any model point  $\mathbf{X}_0$ :

$$\mathcal{L}^{\mathbf{X}_0}(\Phi|\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^N \sum_{(L_m, m) \in \mathcal{D}^i} \sum_{j=1}^K \|g_{i \leftarrow i+j} \mathbf{X}_0^{i+j} - \mathbf{X}_0^i\|^2. \quad (7)$$

where  $\mathcal{D}^i$  denotes LiDAR-mask pairs detected in the frame  $i$ ,  $K$  is the length of the frame sequence over which the consistency is evaluated, and  $N$  is the total number of frames.

Note that loss (7) is null for all object keypoints if, and only if,  $g^i = g_{i \leftarrow j} g^j$ , which is an *equivariance* condition for the predictor. However, we found it more convenient and robust to enforce this equivariance by using (7) summed over a small set of representative model points. In our experiments we have found that at least two points are required to ensure that we consistently predict the heading (yaw) of the detected car. In practice, we use car centre and front keypoints (see Fig. 6), so the final loss term used to train the model is:

$$\mathcal{L}(\Phi, \Psi|\mathcal{D}) = \mathcal{L}'(\Phi, \Psi|\mathcal{D}) + \mathcal{L}^{\mathbf{X}_{\text{centre}}}(\Phi|\mathcal{D}) + \mathcal{L}^{\mathbf{X}_{\text{front}}}(\Phi|\mathcal{D}) \quad (8)$$

**a) Implementation details of tracking:** A detector such as Mask R-CNN only provides 2D mask for individual video frames, but defining the loss (7) requires to identify or track the same object across two frames. For tracking, we take the median of LiDAR points for each mask and compare them to the medians of masks in adjacent frames. The closest pseudo-centres are chosen to be the same vehicle if and only if the number of LiDAR points has not changed significantly and the distance between them is less than 2 meters when ego motion is accounted for. The distance criterion ensures that the same vehicle is detected while the number of points removes poor Mask-RCNN detections in subsequent frames. This tracking is only used at training time, at test time we use all 2D detections.

#### IV. EXPERIMENTS

We assess our methods against the relevant state of the art on standard benchmarks.

**a) Data:** For our experiments, we use the KITTI Object Detection dataset [7] which has 7481 frames with labels in 3D.

However, we do so for compatibility with prior work. Specifically, we use the split found in [3], the standard across all prior works which first splits the videos into training and validation sets focusing on two different parts of the world (no visual overlap). The network is learned on the training videos using multiple frames and then applied and evaluated on the validation videos on a single-frame basis.

KITTI evaluates vehicle detectors using Birds Eye View (BEV) IOU and 3D box IOU (3D) with a strict cutoff of 0.7 for a positive detection. To be compatible with other relevant works in automated labelling [44], [29], we evaluate instead at a threshold of 0.5, also used for other KITTI categories, which reduces the influence of object size on IoU performance. In part, this is motivated by [5] which notes that the size of the ‘ground-truth’ KITTI annotations are often imprecise due

to the fact that many object instances have few LiDAR points and thus it is difficult for human annotators to accurately estimate metric size.

**b) Data preprocessing:** To construct our training set we first run Mask-RCNN [9], [40] with the ResNet 101 backend to locate cars in the images. We then extract the LiDAR points contained within the masks detected and use these for 3D labelling. When tracking for 5 frames this gives us 9.6k cars in the training. For evaluation we only use a single frame of Mask R-CNN detection and use the mask score as the confidence for our predictions.

**c) Implementation:** We implement our method in Pytorch [23] and use components from Frustum PointNets [27] to construct our network. All variants are trained with Adam optimizer with a learning rate of  $3 \times 10^{-3}$  decreasing every 30 steps with multiplier 0.3 for a total of 150 epochs with a batch size of 64. Training was performed on a single Titan RTX with a Ryzen 3900X processor and the most complex models had a training time of 7 hours.

##### A. Ablations of model components

We first start by analysing the individual components of our model (table I). First, we do not use the 2D mask to filter LiDAR points, significantly increasing the number of outliers, arising in particular from cars proximal to the target one (row (a)).

Masking LiDAR points (row (b)) results in a substantial improvement, removing most of these outliers (see also Fig. 3). Introducing the temporal consistency/equivariance loss (7) (row (c)) does not give by itself a noticeable benefit because outlier points are still heavily influential to the prediction. Discounting outliers using the probabilistic formulation of (3) increases accuracy substantially (row (d)). Furthermore, bringing back the consistency loss, which amounts to our full model, does now show a significant benefit (row (e)). Our interpretation is that considering multiple frames can significantly aid discovering and learning outlier patterns: this is because outliers tend to be *inconsistent* across frames, so reasoning over multiple frames helps discovering them (see Fig. 5).

TABLE I: Ablation of different components of the model and data processing steps. Our full model with automatic outlier discounting and multi-view consistency achieves the highest accuracy.

	Filtering	Components		AP <sub>BEV</sub> (IoU = 0.5)		
		Outlier-aware	Multi-view	Easy	Moderate	Hard
(a)	2D Box			35.53	41.54	33.96
(b)	2D Mask			58.61	62.56	54.20
(c)	2D Mask		✓	58.63	61.70	54.11
(d)	2D Mask	✓		75.46	76.60	68.59
(e)	2D Mask	✓	✓	<b>80.73</b>	<b>81.70</b>	<b>73.61</b>

**a) Yaw estimation:** In table II we evaluate our approach for estimating camera viewpoint proposed in section III-C. First we experiment with a network  $\Phi$  tasked to output a vector  $\mathbf{x} \in \mathbb{R}^2$  with the yaw angle computed as  $\theta =$



$\arctan(x_1/x_2)$  (row (a)). Our direct prediction approach (rows (d-g)) outperforms this naïve baseline by a significant margin. Our method is influenced by the number of discrete rotations, 64 being optimal (row (f)). In order to provide stronger baselines, we additionally implement two alternative techniques [10], [8] to handle ambiguous predictions in 3D pose estimation, but did not observe a benefit (rows (b) and (c)) compared to simple direct arctan regression. This is perhaps due to the different setting ([10], [8] were proposed to handle ambiguous fitting of 3D shapes to 2D silhouettes).

TABLE II: Comparison of yaw prediction techniques.

Paradigm		AP <sub>BEV</sub> (IoU = 0.5)		
		Easy	Moderate	Hard
(a)	arctan	76.65	79.05	69.33
(b)	Insafutdinov & Dosovitskiy [10]	77.28	76.92	69.12
(c)	Goel et al. [8]	76.49	77.35	69.69
(d)	Ours, 16 bins	77.04	77.30	69.49
(e)	Ours, 32 bins	79.93	81.14	73.15
(f)	Ours, 64 bins	<b>80.73</b>	<b>81.70</b>	<b>73.61</b>
(g)	Ours, 128 bins	81.79	82.23	74.11

**b) SoTA comparison:** In table III we compare our method to the relevant state of the art. VS3D [29] uses a viewpoint estimation network pretrained on Pascal 3D and NYC 3D cars [42], [18] with ground truth yaw annotations. In Zakharov et al. [45] a synthetic dataset is used to initialise a coordinate shape space NOCS [38] providing a strong prior on 3D shape and yaw estimation. During fitting stage they only utilise Mask R-CNN predictions which have a high IOU compared to a ground truth 2D box and also takes 6 seconds to infer a single car on a modern GPU making it infeasible for real time prediction unlike VS3D [29] (22Hz) and our method which runs at 200Hz after the 2D object detection method (about 25Hz). We provide results (c) with a 2D detector trained on MS-COCO[16] for fairest comparison to this method which has pretrained requirements closest to our own. Frustum PointNet [27] is a fully supervised (2D box, yaw, 3D size, 3D centre) method trained on KITTI that serves as a reference with similar architecture to our method.

**c) Auto-Label Generation:** In table IV we compare different supervision options on a single model. We train the FPN [27] using original 3D ground truth, as well as only the 3D ground truth where there is a matching 2D detection from Mask-RCNN [40] (the same detector used in our method) to demonstrate what is the impact of instances missed by the 2D detector to the overall performance. We

TABLE III: Object Detection Average-Precision on the KITTI validation set. Compared to our Method VS3D[29] uses a network trained on Pascal 3D and NYC 3D Cars[42], [18] to determine the object Yaw and 2D box, while while Zakharov[44] only considers MASK R-CNN detections with an IOU > 50% compared to a ground truth box and uses a synthetic dataset to train a network which gives yaw.

Method	Annotations Source			AP <sub>BEV</sub> / AP <sub>3D</sub> (IoU = 0.5)		
	2D Boxes	3D Yaw	3D Boxes	Easy	Moderate	Hard
(a) VS3D [29]	Pascal 3D	NYC Cars		74.5/40.32	66.71/37.36	57.55/31.09
(b) Zakharov et al. [44]	KITTI	KITTI		77.84/62.25	59.75/42.23	-/-
(c) <b>Ours</b>	MS-COCO			80.73/76.73	81.70/76.66	73.61/69.01
(d) <b>Ours</b>	Cityscapes			<b>86.52/83.45</b>	<b>86.22/79.53</b>	<b>75.53/71.01</b>
(e) <i>Frus.PointNet</i> [27]	KITTI	KITTI	KITTI	98.16/97.67	94.80/93.81	87.11/86.14

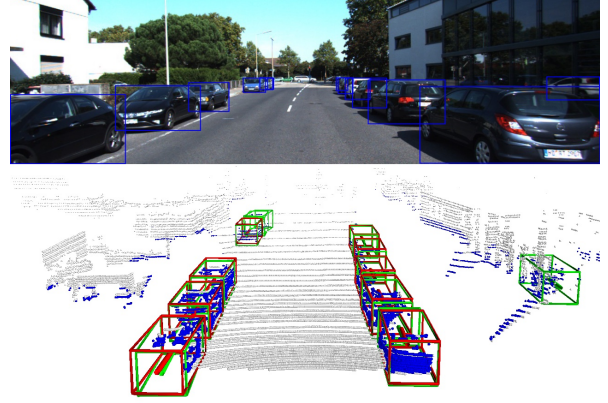


Fig. 6: Qualitative results of our method (green) on the KITTI Validation Set with ground truth car annotations in red and points inside each mask in blue.

show that the difference between our labelling method, which does not use any 3D ground truth, and using full 3D ground truth of only the instances that are “made available” by the 2D detector, is relatively small, and therefore the 2D detection quality is the main limiting factor.

TABLE IV: Using different forms of supervision to train Frustum PointNet [27]

Supervision	AP <sub>BEV</sub> (IoU = 0.5)		
	Easy	Moderate	Hard
Fully Supervised (original GT)	98.16	94.80	87.11
Fully Supervised (GT w/ matching dets.)	91.83	87.16	78.29
<i>Supervised by our labels (no GT)</i>	90.23	85.74	76.84

## V. CONCLUSIONS

In this paper we presented a novel method of localising 3D objects in LiDAR point clouds, trained using only generic 2D object detector. Compared to previous work, our method is less complex, we do not require the use of additional manually annotated data sources as in [29] or virtual data and ground truth 2D bounding boxes as in [45], and we still achieve superior accuracy and better run time. To our knowledge, our method is the first method that can learn to transform 2D predictions to 3D detections without any need for supervision of 3D parameters.

## ACKNOWLEDGEMENT

We are very grateful to Continental Corporation for sponsoring this research. Lukas was supported by OP VVV funded project CZ.02.1.01/0.0/0.0/16019/0000765 “Research Center for Informatics”.

## REFERENCES

- [1] Lucas Beyer, Alexander Hermans, and Bastian Leibe. Biternion nets: Continuous head pose regression from discrete training labels. In *German Conference on Pattern Recognition*, pages 157–168. Springer, 2015.
- [2] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.
- [3] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.
- [4] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1355–1361. IEEE, 2017.
- [5] Di Feng, Lars Rosenbaum, Fabian Timm, and Klaus Dietmayer. Labels Are Not Perfect: Improving Probabilistic Object Detection via Label Uncertainty. *arXiv e-prints*, page arXiv:2008.04168, August 2020.
- [6] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, June 2018.
- [7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [8] Shubham Goel, Angjoo Kanazawa, and Jitendra Malik. Shape and viewpoint without keypoints. In *Proc. ECCV*, 2020.
- [9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *Proc. ICCV*, 2017.
- [10] Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, pages 2804–2814, Montréal, Canada, 2018. Curran Associates.
- [11] Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? *Proc. NeurIPS*, 2017.
- [12] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.
- [13] L. Koestler, N. Yang, R. Wang, and D. Cremers. Learning monocular 3d vehicle detection without 3d bounding box labels. In *Proceedings of the German Conference on Pattern Recognition (GCPR)*, 2020.
- [14] Shuai Liao, Efstratios Gavves, and Cees GM Snoek. Spherical regression: Learning viewpoints, surface normals and 3d rotations on n-Spheres. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9759–9767, 2019.
- [15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
- [16] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *Proc. ECCV*, 2014.
- [17] Francisco Massa, Renaud Marlet, and Mathieu Aubry. Crafting a multi-task CNN for viewpoint estimation. *arXiv preprint arXiv:1609.03894*, 2016.
- [18] Kevin Matzen and Noah Snavely. Nyc3dcars: A dataset of 3d vehicles in geographic context. In *Proc. Int. Conf. on Computer Vision*, 2013.
- [19] Qinghao Meng, Wenguan Wang, Tianfei Zhou, Jianbing Shen, Luc Van Gool, and Dengxin Dai. Weakly supervised 3D object detection from LiDAR point cloud. In *ECCV*, 2020.
- [20] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3D bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017.
- [21] David Novotný, Diane Larlus, and Andrea Vedaldi. Learning 3D object categories by looking around them. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.
- [22] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.
- [24] Hugo Penedones, Ronan Collobert, Francois Fleuret, and David Grangier. Improving object classification using pose information. Technical report, Idiap, 2012.
- [25] Bojan Pepik, Michael Stark, Peter V. Gehler, and Bernt Schiele. Teaching 3D geometry to deformable part models. In *Proc. CVPR*, 2012.
- [26] Sergey Prokudin, Peter Gehler, and Sebastian Nowozin. Deep directional statistics: Pose estimation with uncertainty quantification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 534–551, 2018.
- [27] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. *arXiv preprint arXiv:1711.08488*, 2017.
- [28] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016.
- [29] Zengyi Qin, Jinglu Wang, and Yan Lu. Weakly supervised 3D object detection from point clouds. In *Proc. ACM MM*, 2020.
- [30] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *Proc. NeurIPS*, 2016.
- [31] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. Learning 3-d object orientation from images. In *2009 IEEE International conference on robotics and automation*, pages 794–800. IEEE, 2009.
- [32] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576, 2015.
- [33] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3D model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015.
- [34] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.
- [35] Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Multi-view consistency as supervisory signal for learning shape and pose prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2897–2905, 2018.
- [36] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *Proc. CVPR*, 2015.
- [37] Dominic Zeng Wang and Ingmar Posner. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems*, volume 1, pages 10–15607. Rome, Italy, 2015.
- [38] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [39] Peng Wang, Xinyu Huang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang. The apolloscape open dataset for autonomous driving and its application. *IEEE transactions on pattern analysis and machine intelligence*, 2019.
- [40] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019.
- [41] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond PASCAL: A benchmark for 3D object detection in the wild. In *Proc. WACV*, 2014.
- [42] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014.

- [43] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.
- [44] Sergey Zakharov, Wadim Kehl, Arjun Bhargava, and Adrien Gaidon. Autolabeling 3D objects with differentiable rendering of SDF shape priors. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [45] Sergey Zakharov, Wadim Kehl, Arjun Bhargava, and Adrien Gaidon. Autolabeling 3D objects with differentiable rendering of SDF shape priors. In *Proc. CVPR*, 2020.
- [46] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

## Chapter 7

# Direct LiDAR-based object detector training from automated 2D detections

This paper was presented at the NeurIPS 2020 Workshop on Machine Learning for Autonomous Driving 2020

---

# Direct LiDAR-based object detector training from automated 2D detections

---

**Robert McCraith**  
Visual Geometry Group  
University of Oxford  
robert@robots.ox.ac.uk

**Eldar Insafutdinov**  
Visual Geometry Group  
University of Oxford  
eldar@robots.ox.ac.uk

**Lukas Neumann**  
Visual Recognition Group  
Czech Technical University in Prague  
neumalu1@fel.cvut.cz

**Andrea Vedaldi**  
Visual Geometry Group  
University of Oxford  
vedaldi@robots.ox.ac.uk

## Abstract

3D Object detection (3DOD) is an important component of many applications, however existing methods rely heavily on datasets of depth and image data which require expensive annotation in 3D thus limiting the ability of a diverse dataset being collected which truly represents the long tail of potential scenes in the wild. In this work we propose to utilise a readily available robust 2D Object Detector and to transfer information about objects from 2D to 3D, allowing us to train a 3D Object Detector without the need for any human annotation in 3D. We demonstrate that our method significantly outperforms previous 3DOD methods supervised by only 2D annotations, and that our method narrows the accuracy gap between methods that use 3D supervision and those that do not.

## 1 Introduction

3D Object Detection in LiDAR data (3DOD) is an important component of many applications, ranging from autonomous cars to robotics. However compared to the more traditional 2D Object Detection, which is an extensively studied field of Computer Vision, its 3D counterpart on the other hand has seen slower development, mainly due to challenging data acquisition and even more challenging human annotation requirements, as labeling objects in 3D is substantially more expensive and more time consuming while also being error prone. As a result in the context of autonomous driving, there is only a handful of annotated datasets Geiger et al. [2012], Caesar et al. [2019], Sun et al. [2020] for 3D Object Detection, but still they only contain data for one or two cities, that reduces their ability to generalise to other locations or to capture rare events (aka the *long tail problem*), which is especially important in such a safety-critical application.

In order to address the aforementioned 3D annotation requirements, we instead propose to exploit a readily available 2D Object Detector on a *image sequence* as a form of weak supervision to create the training signal for the 3D Object detector in *LiDAR data*, thus completely removing the need for 3D human annotations. The cross-modal training approach has several advantages: Firstly, image object detection is a well-studied problem with robust methods on very diverse data with minimal bias towards specific models of car which may be more prominent in some locations. Secondly, by only requiring a 2D detection at training time, we reduce drastically the amount of annotation required to label new data. And last but not least, transferring detections between domains means that our network must learn to reason about the 3D shape of objects, rather than presuming that an object is present, which can cause problems for 3DOD methods which rely on image-based detectors at test time as well Zakharov et al. [2020a], McCraith et al. [2022].

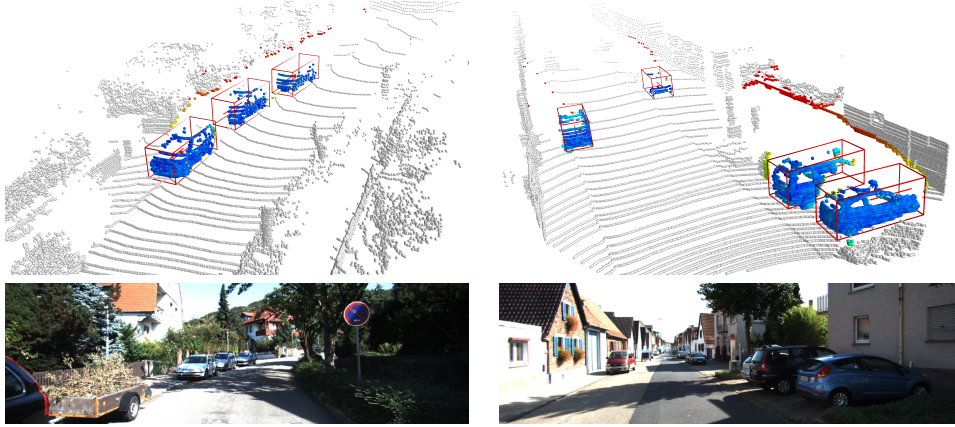


Figure 1: Qualitative examples of our LiDAR 3D Object Detection method. LiDAR point cloud and detected cars (top row) and scene image (for visualization purposes only - bottom row)

In this paper, we introduce three key contributions to weakly-supervised 3DOD training: i) Unlike previous methods, the 2D Object Detector is only required at training time, which not only simplifies the pipeline and removes need for additional sensors in the car or a robot, but it also eliminates sensitivity to missed detections by the image object detector, which is especially prevalent for small far-away objects. ii) We introduce a new *Soft Inlier Count* (SIC) loss function which allows us to train the whole system end-to-end using standard back-propagation. iii) A new multi-cell voting scheme is introduced to relax the original hard-choice training loss of the fully-supervised method, allowing the model to gradually improve its estimates as the training progresses.

## 2 Related

### 2.1 Supervised 3D Object Detection

Image object detection methods are quite mature, which resulted in many early 3D object detection methods making detections in RGB and using these to select LiDAR data on which to reason about 3D location. A notable example of this is Frustum PointNets Qi et al. [2017]. In this method an image based detector produces a set of bounding boxes whose frustums are used to select a subset of the LiDAR point cloud. These points are then segmented to further reduce down to only points inside the 3D bounding box and a final stage produces location and rotation estimates.

Another popular technique is to perform detection and localisation on LiDAR data alone. A method popularised in Zhou and Tuzel [2018] is to gather LiDAR points which fall into a voxel and learn features voxel-by-voxel, then take these features and organize them to construct a BEV image of features upon which further convolutional layers derive location. This has been expanded on in Lang et al. [2019], Yin et al. [2021] where voxels are expanded in the vertical axis to contain all points in a square patch (in BEV) and the resulting prediction are decoded by anchors or heatmaps.

### 2.2 Weakly Supervised 3D Object Detection

Recent works have attempted to resolve annotation difficulties by utilising a coarse annotation strategy for a large set with fine annotations for a smaller set thus making annotation less time consuming/costly. In Meng et al. [2020] they use a small number of weak annotations of bounding box centres in BEV and a small amount of exact 3D box annotations. A detector is trained in a two-stage manner by first predicting object proposals in BEV and then regressing accurate 3D localisation. Such a coarse labeling strategy also can result in some of the problems mentioned in Feng et al. [2020] such as inability to accurately localise car center under heavy occlusion, partial scans of objects based on their perceived orientation and a poor ability to determine object shape from some viewpoints. Qi et al. [2021] run a pre-trained detector (on a small subset of labeled data) on an entire LiDAR sequence to produce a set of 3D boxes which are then passed into a multi-target tracker. Utilising complimentary information from multiple frames allows them to improve detection accuracy and apply their auto-labeler in a semi-supervised scenario.

**Cross-modal supervision.** Other formulations of this problem utilise additional datasets to supervise components of 3D detection pipelines. Qin et al. [2020] trains a 3D object proposal network based on distance-normalised point cloud density. In the second stage, an image-based network pretrained on a different dataset to classify proposals and predict viewpoint image bounding box is used as a teacher for a LiDAR-only model. An alternative approach is used in Zakharov et al. [2020b] where a network is pretrained on Parallel Domain McNamara [2020] – a synthetic dataset to provide a good initial estimate of translation and rotation. This network then takes Mask R-CNN He et al. [2017] detections that have a high overlap with a ground truth bounding box to refine the parameters on read data. Finally, McCraith et al. [2022] also takes Mask R-CNN detections and uses a Frustum PointNet style architecture and attempts to learn which points are inliers to remove noise, as with the other methods however this method depends on the availability of camera and LiDAR at test time (and their coordinate transformation remains consistent). Compared to these approaches we rely on the same 2D detection but only a training time. This has an advantage that our trained detector operates on LiDAR input and can recover objects that an RGB-based detector may fail to predict, for example when the car is reflected from a glass wall. Training a detector end-to-end allows to exploit regularities in the input LiDAR space and effectively utilise available 2D supervision.

### 3 Method

Our goal is to determine the 3D bounding boxes without the need for expensive, time consuming human annotation which severely limits the construction of large scale datasets for 3DOD. To achieve this we leverage mature 2D object detection methods for which the annotation is much simpler and more large scale datasets with a diverse set of data exist from around the globe rather than a small area as is typically seen in 3D Object Detection datasets.

Cross modal supervision such as this however has several challenges. 2D instance segmentation in pixel space is a crude approximation of segmentation in 3D point clouds, with many pixels at the boundary of objects being mis-classified (or coming from the region in the objects frustum from one sensor but not the other), the transparency of windows results in LiDAR points correctly attributed to a vehicle in image space to actually be located on surfaces behind the vehicle, and in many cases LiDAR scans may only have very small parts of a vehicle visible leading to ambiguity or orientation and translation.

3D Object Detection is typically reduced to regressing values for  $(X, Y, Z)$  center,  $(l, w, h)$  size and  $\theta$  yaw, rather than the other option of predicting the 3D location of the 8 corners of the 3D bounding box. In many cases object size is a hard to determine quantity, as noted in Feng et al. [2020] in many cases only one side of a car is visible making annotations for size unreliable, with this in mind we use a generic car shape with a fixed size to train out model to predict location and orientation which we feel are much more important quantities for downstream tasks.

#### 3.1 Exploiting 2D Labels in a 3D Point Cloud

In our method, the training signal is generated by an off-the-shelf 2D object detection and segmentation system, such as Mask R-CNN He et al. [2017]. Given an RGB image of the scene  $I \in \mathbb{R}^{3 \times H \times W}$ , the 2D detector creates a set of detections  $D$  in the form of a segmentation mask  $m$  for each detected car

$$D = \{m \in \{0, 1\}^{H \times W}\} \quad (1)$$

Taking a corresponding LiDAR point cloud of the same scene  $L = \{(x, y, z, r) \in \mathbb{R}^4\}$ , we define the point cloud subset  $L_m$  for each detection  $m$  as

$$L_m = \{p \in L : \text{proj}(p) \in m\} \quad (2)$$

$$\text{proj}(u) = K T_{\text{cam}} u \quad u \in \mathbb{R}^3 \quad (3)$$

where  $\text{proj}(u)$  is the projection of the world point  $u$  onto the 2D image given by known camera intrinsics  $K$  and a transformation from LiDAR to camera co-ordinate system  $T_{\text{cam}} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ .

While in some cases it may be possible to directly reconstruct the 3D shape of an object given enough LiDAR points exist in the  $L_m$  subset, in our context the object is only typically observed from one side, resulting in partial scans at best. This is especially a problem for cars whose apparent angle (how it appears in the image rather than its global orientation) suggests that the object is moving

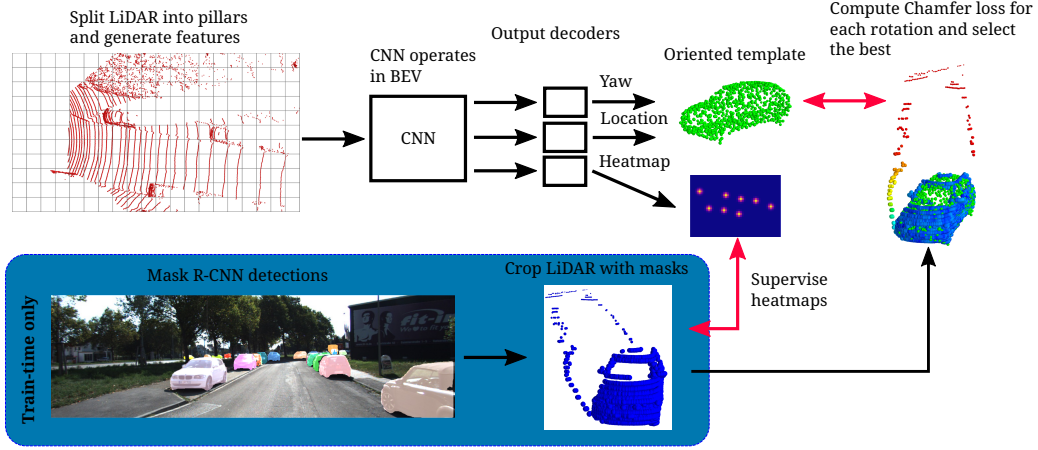


Figure 2: Our method only uses an off-the-shelf 2D object detector (bottom) to train a LiDAR 3D object detector (top), yet still achieving comparable accuracy to full supervision by laborious and costly 3D annotations.

away from the ego vehicle. As a result of this ambiguity we choose to utilise a 3D rigid model  $M$  of a car with a typical car size and shape, which we fit to observed LiDAR points  $L_m$ . The model  $M = \{p \in \mathbb{R}^3\}$  is translated to a world location and rotated using the translation  $T$  and orientation  $\theta$ , assuming that the translation and the orientation is assumed to be the car position and rotation (yaw) respectively.

We then define the distance measure between the translated model  $\mathcal{T}_{T,\theta}(M)$  and the observed object point cloud  $L_m$  as

$$d(L_m, M \mid T, \theta) = \frac{1}{|L_m|} \sum_{p \in L_m} \min_{p' \in \mathcal{T}_{T,\theta}(M)} \|p - p'\|^2 \quad (4)$$

This distance measurement is similar to Chamfer distance which measures the distance between point clouds  $A$  and  $B$  by taking the closest point in set  $B$  for each point in  $A$  and vice versa, however in our case we only measure distance for observed every point in  $L_m$  as measuring distance also for every model point  $M$  would not work when only part of the object is captured as is often the case.

Note that the distance in Eq. 4 is fully differentiable with respect to the translation/rotation parameters  $T$ , and therefore it might be natural to ask if we could just iteratively search locations and compute this metric then picking the location which minimizes the distance. This however results in poor performance as outlier LiDAR points shift our prediction away from the expected location and when the available points represent a small section of the the target vehicle this distance can be unstable.

We therefore propose sharing information on object locations across frames by using a deep neural network  $\Psi$  which takes a LiDAR point cloud of the whole scene and generates a set of object detections  $\Psi : L \rightarrow \{(T_i, \theta_i)\}$ , where each detected object  $i$  is encoded by a 3D position of its center  $T_i$  and an orientation  $\theta_i$ . We then define a loss  $\mathcal{L}$  for one scene as

$$\mathcal{L}(\Psi \mid L, D, M) = \frac{1}{|D|} \sum_{m \in D} d(L_m, M \mid T_m, \theta_m) \quad (T_m, \theta_m) \in \Psi(L) \quad (5)$$

and we train the network  $\Psi$  by optimizing the loss over the whole dataset.

### 3.2 Soft Inlier Count metric

While sharing knowledge of locations across frames allows for a more robust estimation with partial point clouds, another issue still exists: often additional LiDAR points not belonging to the car are mistakenly included in the  $L_m$  subset, which skews the distance measurement and the resulting



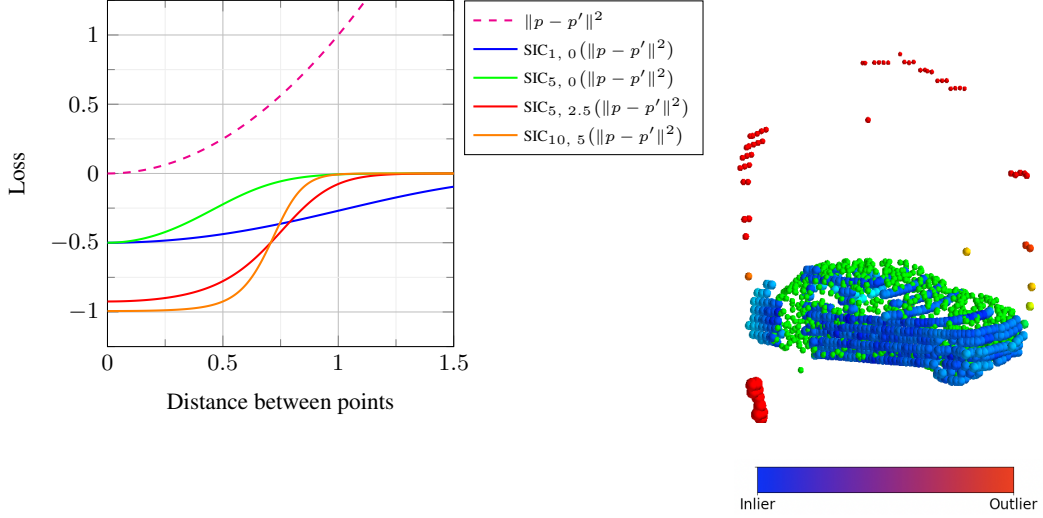


Figure 3: *Soft Inlier Count (SIC) loss*. Using L2 norm directly results in outlier points having a disproportionate influence on the loss minimising location. Different parametrisations of the SIC loss (left). An example matching of the 3D rigid model (green) to the detected object point cloud, with SIC loss value illustrated by the colour of each LiDAR point (right).

location/rotation estimate. Even the best 2D Object Detectors will make some erroneous predictions at a pixel level, caused by occluders of similar appearance, or LiDAR not reflecting properly. These points have a large value in our loss function for a correctly positioned model  $M$  which results in the outliers pulling the prediction away from the correct location even if they are small in number (see Fig. 3 - dashed line).

To address the outlier issue, we propose a new *Soft Inlier Count (SIC)* loss to soften the L2 metric of Eq. 4 with a sigmoid function as

$$\bar{d}(L_m, M \mid T_m, \theta_m) = \frac{1}{|L_m|} \sum_{p \in L_m} \sum_{p' \in \mathcal{T}_{T_m, \theta_m}(M)} \text{SIC}_{\alpha, \beta}(p, p') \quad (6)$$

$$\text{SIC}_{\alpha, \beta}(p, p') = -\frac{1}{1 + \exp(-\alpha \|p - p'\|^2 + \beta)} \quad (7)$$

where  $\alpha$  and  $\beta$  are method parameters whose value is determined empirically (see Sec. 4.2) and again  $\mathcal{T}_{T, \theta}(M)$  denotes the rigid model  $M$  translated by  $T$  and rotated by  $\theta$ .

As a result, instead of using the raw L2 distance between the LiDAR and template rigid model we maximize the number of points sufficiently close to the template at a given location, with the assumption that most of the LiDAR points in the mask fall on the object we are interested in. We then sum across the entire set of LiDAR points  $L_m$  selected by the 2D Object Detector, which gives a soft count of how many points LiDAR points are close to the model  $M$  as well as giving an idea of quality, while still being fully differentiable.

### 3.3 Model Architecture

Our model is based on the CenterPoint architecture Yin et al. [2021], where LiDAR points are first separated into voxels of infinite height, then are fed through a network to generate features for this spatial volume. These feature maps are then scattered into a sparse Birds Eye View map of features upon which a CNN is run. We make two notable improvements to the CenterPoint architecture:

**Multi-cell Voting Scheme.** The original CenterPoint architecture uses the 3D ground truth center location to supervise the center heatmap head, where it selects a single cell of the heatmap as the positive target while the rest of the heatmap is taken as negative. This hard choice makes sense when 3D ground truth is available, however in our case as the 3D information is unavailable and therefore we need to take into account that the initial estimate of the object center might be well off, especially when the object point cloud is incomplete as this will skew the center estimate towards the visible

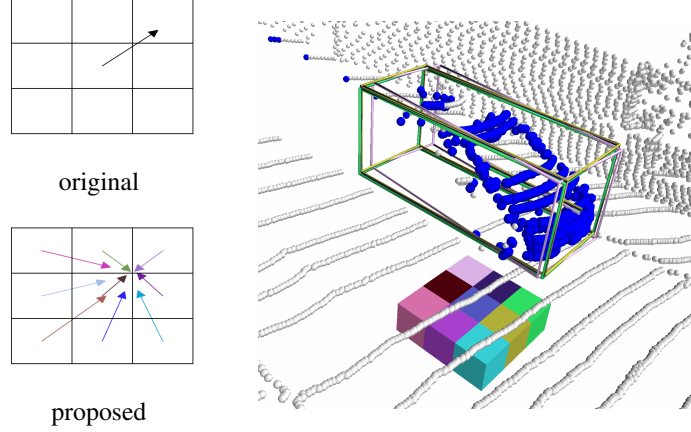


Figure 4: *Multi-cell Voting Scheme*. Rather than making a single hard prediction (top-left), our method produces multiple hypotheses by also making predictions in neighbouring cells of the feature map (bottom-left). A sample scene point cloud with the neighbouring cells and their corresponding predicted 3D bounding-boxes (right); colour of each bounding box encodes the source cell which produced it.

part. To mitigate this issue instead of making a hard choice of center we instead initially pick the median of the object point cloud and use this to select a neighbourhood of cells (see Fig. 4). During a forward pass the prediction made by each of these cells is evaluated and the center of the best prediction (= the prediction with the lowest loss value) is used as an updated center.

**Yaw Estimation.** Estimating rotation angle of the object (= yaw) is a surprisingly complex task with a selection of methods to predict such quantities available. Regression is problematic as a prediction near  $-\pi$  for an object with orientation near  $\pi$  causes a loss value much higher than the actual error. CenterPoint Yin et al. [2021] instead predicts a vector  $(\sin \theta, \cos \theta) \in [-1, 1]^2$  to address this boundary issue. Whilst such a formulation works well if the target (true) orientation is known at training time, when the ground truth is not available inherent ambiguities create deep local minima of the loss function which make the training unstable. The most common ambiguity is caused by the fact that for partial object point cloud all four  $90^\circ$ -step rotations typically have a very similar loss.

In order to help the model not to get stuck in these local optima, at training time in every iteration we instead forcibly try all possible orientations and pick the orientation with lowest loss value as the target. To make this computationally feasible, we quantise the space of all orientations  $[-\pi, \pi]$  into 64 distinct bins and transform the yaw prediction into a 64-way classification. More formally, the final loss function  $\tilde{\mathcal{L}}(\Psi \mid L, D, M)$  is

$$\theta_m^* = \underset{\theta' \in \mathcal{R}}{\operatorname{argmin}} \bar{d}(L_m, M \mid T_m, \theta') \quad (8)$$

$$\tilde{\mathcal{L}}(\Psi \mid L, D, M) = \frac{1}{|D|} \sum_{m \in D} \left( d(L_m, M \mid T_m, \theta_m^*) + \operatorname{CE}(\theta_m \mid \theta_m^*) \right) \quad (T_m, \theta_m) \in \Psi(L) \quad (9)$$

where  $\mathcal{R}$  represents the set of possible rotations and CE denotes the standard cross-entropy loss.

To summarize, our model takes a set of point cloud detections  $\{(x, y, z, r) \in \mathbb{R}^4\}$  as the input (where  $r$  denotes the reflectance value) and it outputs a dense feature map  $\mathcal{M} \in \mathbb{R}^{H \times W \times F}$ , consisting of a center heatmap head  $[H \times W \times 1] \in (0, 1)$ , a regression head  $[H \times W \times 3] \in \mathbb{R}^3$  encoding the object center 3D location offset from the corner of the cell, and the orientation head  $[H \times W \times 64] \in \mathbb{R}^{64}$ . The size of the feature map is  $H = 200$  and  $W = 176$ , therefore one cell corresponds to the size of 40cm in world coordinates when 10cm voxel edge length is used (as the convolutional network downsamples 4 times).

## 4 Experiments

### 4.1 Training and Inference

The KITTI Object Detection dataset Geiger et al. [2012] has 7481 publicly available 3D annotated frames. This is usually split into roughly equal-sized training and validation sets with no sequence of frames present in both originally created in Chen et al. [2017]. This allows us to compare to both fully-supervised existing methods and also the weakly-supervised works Zakharov et al. [2020b], Qin et al. [2020], McCraith et al. [2022] who only present results on this dataset. The standard evaluation in Birds Eye View (BEV) IoU with a strict threshold of 70% is used.

To prepare our dataset Mask R-CNN He et al. [2017], Wu et al. [2019] is used to retrieve objects in image space and generate the masks of point clouds which each detection will be compared to. The entire collection of points is augmented with rotation, translation, left-right flipping and scaling. The points are then sorted into the relative voxel bins which are fed into the network. Each LiDAR mask is compared to the regression and yaw classifier outputs of the corresponding output cells. During inference only the LiDAR data is utilised and detection is performed by the predicted heatmap with Non-maxima Supression (NMS) to prevent each nearby cell predicting the same car.

To evaluate our method we compare to all relevant weakly-supervised 3DOD methods that provide results on the KITTI dataset (see Tab. 1). Compared to McCraith et al. [2022], our method performs significantly better and it only requires the availability of LiDAR at test time, the benefit of this being that our detector is forced to reason about the shape of cars in the input LiDAR, rather than simply attempting to fit the part of the frustum with the greatest density. Zakharov et al. [2020b] pre-train their network on synthetic data and have a slow optimisation step for each instance detected by Mask R-CNN. Because of this they use their method to generate autolabels and feed these into PointPillars Lang et al. [2019], a very similar architecture to CenterPoint Yin et al. [2021], however they ignore more difficult image detections and still struggle with high IoU thresholds. For reference and to evaluate the gap between full and weak supervision, we also include results using the same underlying backbone Yin et al. [2021] as our method. We observe that compared to previous weakly-supervised methods, the gap is significantly smaller. We also observe that when removing the prediction of car size our method is remarkably close to a fully supervised network in the easy case, with a drop off in the moderate and hard cases, likely owing to the often extremely low number of LiDAR Points available for such car instances making it difficult for our inlier counting method to determine a meaningful fit.

### 4.2 Ablations

**Soft Inlier Count Loss.** To evaluate how well our mesh describes the observed point cloud we use must ensure the locations predicted result in an object at locations where LiDAR has collected data and are determined to belong to a car using the image mask. Chamfer distance or its one-sided variant are commonly used to compare point clouds, but using the standard L2 loss is problematic as presence of outlier points will disproportionately change the location of the associated vehicle, resulting in suboptimal performance (see Tab. 2 - first row). By using the newly introduced Soft Inlier Count (SIC), we observe this quantitatively in Tab. 2 that the accuracy is better, and we found that although

Method	Supervision	AP <sub>BEV</sub> (IoU = 0.7)		
		Easy	Moderate	Hard
McCraith et al. [2022] <sup>†</sup>	2D detections	66.7	64.7	57.9
Zakharov et al. [2020a]	2D detections + synth 3D	81.00	59.80	-
<b>ours<sup>†</sup></b>	2D detections	<b>86.39</b>	<b>74.79</b>	<b>67.31</b>
CenterPoint <sup>†</sup> Yin et al. [2021]	3D ground truth	88.08	83.41	80.72
CenterPoint Yin et al. [2021]	3D ground truth	90.84	84.45	82.30

Table 1: Weakly-supervised 3D Object Detection accuracy on KITTI validation set. Fully-supervised CenterPoint Yin et al. [2021] included for reference, methods using average 3D bounding-box instead of predicting actual size denoted with <sup>†</sup>. 2D detection networks are trained on Cityscapes Cordts et al. [2016] (McCraith et al. [2022]) or COCO Lin et al. [2014] (Zakharov et al. [2020b])

the SIC loss is not extremely sensitive to specific parameter settings, the best parameter values were determined to be  $\alpha = 5$  and  $\beta = 0$ . We also observe that for high  $\alpha$  values the loss becomes too strict for our use case, as it becomes impossible to precisely match our rigid one-size-fits-all car model to actual LiDAR detections.

Loss Function	AP <sub>BEV</sub> (IoU = 0.7)		
	Easy	Medium	Hard
$\ p - p'\ ^2$	69.14	61.64	52.11
SIC <sub>1, 0</sub> ( $\ p - p'\ ^2$ )	79.23	67.26	59.69
SIC <sub>5, 0</sub> ( $\ p - p'\ ^2$ )	80.17	69.58	63.73
SIC <sub>5, 2.5</sub> ( $\ p - p'\ ^2$ )	79.91	69.88	63.19
SIC <sub>10, 5</sub> ( $\ p - p'\ ^2$ )	72.94	64.08	56.26

Table 2: Ablation of the *Soft Inlier Count* (SIC) loss with different parameter values and its comparison to the standard L2 loss on the KITTI validation set.

Window Size	AP <sub>BEV</sub> (IoU = 0.7)		
	Easy	Medium	Hard
0	80.17	69.58	63.73
1	84.88	74.53	67.16
2	86.39	74.79	67.31
3	82.05	72.26	66.58

Table 3: Ablation of the multi-cell voting scheme. Only a single heatmap cell (window size = 0) as in CenterPoint Yin et al. [2021] performs worst and too big window sizes create ambiguities, eg. when cars are parked closely.

**Multi-cell Voting.** Normally in anchor/location based predictions such as PointPillars Lang et al. [2019] and CenterPoint Yin et al. [2021] the object center is known and can be used to assign a heatmap location and choose which the corresponding cell in the predictions to penalise in order to calculate the loss. In our case however the center is unknown, which means some approximation must be used. Regardless of how this center is chosen as it can only be based on the partial captures of the LiDAR is will inherently be biased in some way, depending on how much of the car is visible/detected by the mask. Quantitatively we observe in Tab. 3 that using only a single cell indeed has sub-optimal accuracy, whilst on the other end using too large window of cells becomes too large and causes ambiguities when objects are placed close together, as now multiple cars can potentially be predicted by the same cell. To summarize, in our method we opt to use window size of 2 (ie.  $5 \times 5$  cells) as it provides decent accuracy boost while being significantly faster to train than larger window sizes.

## 5 Conclusion

In this work, we utilised a readily available robust 2D Object Detector to transfer information about objects from 2D to 3D, allowing us to train a 3D Object Detector without the need for any human annotation in 3D. We demonstrated that our method significantly outperforms previous 3DOD methods supervised by only 2D annotations, and that our method narrows the accuracy gap between methods that use laborious and costly 3D supervision and those that do not.

## Acknowledgement

We are very grateful to Continental Corporation for sponsoring this research. Lukas was supported by OP VVV funded project CZ.02.1.01/0.0/0.0/16019/0000765 “Research Center for Informatics”. Robert gratefully acknowledges support from the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines & Systems.

## References

- Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.
- Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

- Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. CVPR*, 2016.
- Di Feng, Lars Rosenbaum, Fabian Timm, and Klaus Dietmayer. Labels Are Not Perfect: Improving Probabilistic Object Detection via Label Uncertainty. *arXiv e-prints*, art. arXiv:2008.04168, August 2020.
- Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *Proc. ICCV*, 2017.
- Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.
- Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
- Robert McCraith, Eldar Insafutdinov, Lukas Neumann, and Andrea Vedaldi. Lifting 2d object locations to 3d by discounting lidar outliers across objects and views. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2411–2418. IEEE, 2022.
- Kevin McNamara. Parallel domain: Data generation for autonomy. <https://paralleldomain.com/>, 2020.
- Qinghao Meng, Wenguan Wang, Tianfei Zhou, Jianbing Shen, Luc Van Gool, and Dengxin Dai. Weakly supervised 3D object detection from LiDAR point cloud. In *ECCV*, 2020.
- Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. *arXiv preprint arXiv:1711.08488*, 2017.
- Charles R Qi, Yin Zhou, Mahyar Najibi, Pei Sun, Khoa Vo, Boyang Deng, and Dragomir Anguelov. Offboard 3d object detection from point cloud sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6134–6144, 2021.
- Zengyi Qin, Jinglu Wang, and Yan Lu. Weakly supervised 3D object detection from point clouds. In *Proc. ACMM*, 2020.
- Pei Sun, Henrik Kretschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.
- Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019.
- Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *CVPR*, 2021.
- Sergey Zakharov, Wadim Kehl, Arjun Bhargava, and Adrien Gaidon. Autolabeling 3D objects with differentiable rendering of SDF shape priors. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, June 2020a.
- Sergey Zakharov, Wadim Kehl, Arjun Bhargava, and Adrien Gaidon. Autolabeling 3D objects with differentiable rendering of SDF shape priors. In *Proc. CVPR*, 2020b.
- Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

# Chapter 8

## Conclusion

### 8.1 Impact

#### 8.1.1 Relative Velocity Estimation

In Chapter 3 we introduced a simple method for real time relative velocity estimation based on a sequence of images. This work not only showed that a simplified representation of the input data (namely as a sequence of bounding boxes) not only performs similarly to complete images but does this with much quicker runtime and allows artificial datasets to be created substantially easier for this task. This allows future solutions to this problem to train on a more diverse range of data, which allows the network generalise to unseen velocities in different environments.

#### 8.1.2 Monocular Depth Estimation with self-supervised instance adaption

Self-supervised monocular depth estimation presents a powerful technique for the estimation of pixelwise depth without the need for an additional supervising sensor at dataset creation time. These methods like many other learned solutions these networks tend to perform better on the training set than on unseen testing data. To help alleviate these issues we explored how the parameters of the network could be modified at test time to improve the outputs without overfitting to the current example in Chapter 4. This is achieved by only modifying specific layers of the network which are most sensitive to the input image, while freezing later layers which contain more abstract knowledge about the depth of objects with a certain scale in an image. Further to this we showed that due to the gradual spatio-temporal changing of driving scenery we can perform fewer steps of gradient descent at each frame as

the parameter updates in one frame can be retained in subsequent frames, meaning our test time adaption strategy can be deployed in near real time on a GPU.

### **8.1.3 Calibrating self-supervised monocular depth estimation**

Self-supervised monocular depth estimates are learned at an undetermined scale. During regular evaluation it is common to perform scaling based on the median ratio of LiDAR depth projected into the image compared to the predicted value. The result of this is that self-supervised monocular depth estimates are difficult to use in practice as getting meaningful values requires a LiDAR onboard, which limits the utility of our camera based methods. In Chapter 5 we propose the use of the camera height which is part of the known sensor calibration to determine the scale of the predicted depth. This is achieved by segmenting the pixels containing only road and extracting a point cloud using the predicted depth image. On this point cloud we fit a plane and derive the pseudo camera height for the networks depth scale. This value can then be compared to the known camera height and the ratio used to scale the monocular depth prediction. This approach results in accuracy similar to that of scaling with LiDAR while not requiring additional sensors unlike [22]. This work was released simultaneously to [36] which utilizes a very similar approach.

### **8.1.4 Lifting 2D Object Locations to 3D by Discounting LiDAR Outliers across Objects and Views**

While monocular depth estimation has seen many previous works, very few papers have attempted the obvious downstream task on 3DOD without full supervision. In Chapter 6 we utilise the popular Frustum PointNet[32] architecture and a pre-trained Mask R-CNN to explore how 3D object detection can be performed without any supervision on the target dataset. To achieve this we first reduce the LiDAR point cloud by selecting only those points which project in to an instance predicted by Mask R-CNN, allowing us to avoid confusion during training in the common case where multiple cars are in the same 2D image bounding box. Next for each point remaining we predict a per point distance which acts as a soft semantic segmentation. We then take the networks predicted location and sample a range of orientations all of which are evaluated by a single sided chamfer distance, with the minimal loss rotation providing a yaw class as well as a gradient to fine tune the location prediction. This strategy is further improved by comparing predictions for the same car across frames

to ensure consistency, reducing problems caused ambiguous point clouds in a single frame. This approach improved substantially the state of the art of weakly supervised 3DOD without requiring the addition of object difficulty, synthetic datasets, lengthy optimisation or additional 3D information as in the previous best works[42, 33]

### **8.1.5 Direct LiDAR-based object detector training from automated 2D detections**

Weakly supervised 3DOD methods depend on 2D object detections which can be a point of strength owing to the wide range of diverse datasets available easily but of weakness resulting due to naturally occurring adversarial sensor readings such as reflections, false positive detections and commonly mislabeled objects from more generic datasets. Indeed with methods such as Chapter 6 have the somewhat unusual approach of performing detection in one modality (images) and localisation in another (LiDAR), which results in false positive detections made in image space resulting in a detection in LiDAR where the sensor observations would easily rule out a detection in the area. With this in mind we explore using the image detections only at training time in Chapter 7. The idea here being that if LiDAR provides the best information for localising 3D Objects then it must also be beneficial to perform detections in this modality with the noisy teacher from the image as a supervisory signal.

## **8.2 Future Work**

### **8.2.1 Relative Velocity Estimation**

The work described in Chapter 3 describes a limited use case of vehicles on a motorway where relative motion is constrained greatly by the road geometry generally being consistent straight or gently curving roads. While our synthetic and the real dataset contained vehicles of a wide variety of sizes the angular velocity of the examples is very limited owing to the road structure. A natural extension of this could be attempting to train a similar model on a more diverse set of environments where vehicles take much sharper turns and road have much more complex structures.

### **8.2.2 Monocular Depth Estimation**

In Chapter 4 we showed that monocular depth estimates could be fine tuned at test time for greater performance. As the longest sequences evaluated was performed on the KITTI odometry dataset where the duration of the videos not exceeding 20



minutes, which is relatively short for many trips taken in a car these methods need exploration on much longer sequences to explore long scale limitations. A federated learning approach to fine tuning could also be explored, for example since the KITTI and many other far more recent datasets were captured new methods of urban transport such as electronic scooters have become common in many cities, with the likely interpretation of these being pedestrians who's expected movements would differ substantially to an electric scooter user. Adapting to images captured in a very different environment would also be a compelling subsequent work, as the Cityscapes trained model adapted to KITTI is ultimately limited by being two datasets captured in the same country, therefore having many similar looking elements in the images. Chapter 5's method causes a noticeable increase in network latency because of a slow surface fitting, following this work we trained a supervised model with the scaled outputs of this work, however the results were less satisfactory, the only approach which yielded somewhat promising results was a self-supervised monodepth trained on stereo pairs which when used as a supervision signal gives results close to an image of interpolated LiDAR (perhaps owing to the semantically aware object boundaries rather than standard interpolation helping make the empty pixels from LiDAR projected into the image more accurate). [36] provides another future direction where a similar scaling technique is used while calculating the loss during training, meaning the network learned scale aware depth representations directly in it's outputs.

### 8.2.3 Weakly Supervised 3D Object Detection

The work in Chapter 6 and Chapter 7 provides a strong baseline for future work in this area. Natural uses of this as is could be to give a good initial estimate of the 3D object's location and having an annotator fine tune. This could even help in the strategies of [30, 5] which rely on a large amount of coarsely annotated 3D Objects and fine tunes at the end of training on a more accurately labeled much smaller dataset. The work described in Chapter 7 could also be expanded to utilise multiple frames which might help improve performance on LiDAR datasets with lower density point clouds. Both works also are limited to only cars, with many other objects, some more challenging such as pedestrians also being of great interest in autonomous driving. Integrating more complex tracking or integrating multi frame multi object training into the training would also likely prove beneficial for object detection.

## 8.3 Conclusion

Over the course of this thesis I have explored three fundamental tasks for autonomous driving and developed methods to tackle these problems without the need for laborious human annotation. Relative velocity estimation is a fundamental requirement of any future planning, with the work presented in this thesis network trained for this task can be more robust to extremely rare cases while retaining highly accurate results in much more typical cases. My work on monocular depth estimation greatly reduces the error in real world usage, pushing the accuracy very close to that seen on the training data in reasonable time. I also presented a method for getting metric depth values from relative predictions, while requiring no additional sensors for training or application, making the predicted values of self-supervised monocular depth estimation more useful for downstream applications. And finally my work on weakly supervised 3D Object Detection enables the conversion of easily available robust 2D detections to 3D bounding boxes, which enables much larger amounts of data to be trained on, removing the limits on dataset scale put in place by the large annotation cost. This is further extended to remove the need for a camera in application and be significantly more accurate at a much stricter threshold.

# Appendix A

## Statements of Authorship

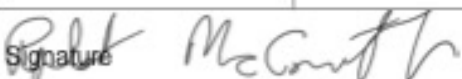
### Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).


Title of Paper	Relative Velocity Estimation
Publication Status	<input type="checkbox"/> Published
Publication Details	McCraith, R., Neumann, L., Vedaldi, A.  IEEE Intelligent Vehicles Symposium 2021

#### Student Confirmation

Student Name:	Robert McCraith		
Contribution to the Paper	performed experiments, wrote code, interpreted data, wrote manuscript.		
Signature 	Date	11/2/23	

#### Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Andrea Vedaldi		
Supervisor comments  This is an accurate description of the contributions.		
Signature 	Date	11/2/23

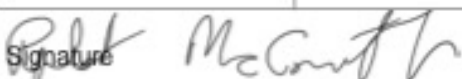
### Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).

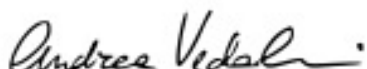
Title of Paper	Monocular Depth Estimation with Self-supervised Instance Adaption
Publication Status	<input type="checkbox"/> Unpublished and unsubmitted work written in a manuscript style
Publication Details	McCraith, R., Neumann, L., Zisserman, A., Vedaldi, A.

#### Student Confirmation

Student Name:	Robert McCraith		
Contribution to the Paper	performed experiments, wrote code, interpreted data, wrote manuscript.		
Signature 	Date	11/2/23	

#### Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Andrea Vedaldi		
Supervisor comments  This is an accurate description of the contributions.		
Signature 	Date	11/2/23

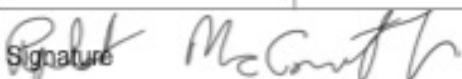
### Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).


Title of Paper	Calibrating Self-supervised Monocular Depth Estimation
Publication Status	<input type="checkbox"/> Published
Publication Details	McCraith, R., Neumann, L., Vedaldi, A.  NeurIPS 2020 Workshop on Machine Learning for Autonomous Driving

#### Student Confirmation

Student Name:	Robert McCraith		
Contribution to the Paper	performed experiments, wrote code, interpreted data, wrote manuscript.		
Signature		Date	11/2/23

#### Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Andrea Vedaldi			
Supervisor comments  This is an accurate description of the contributions.			
Signature		Date	11/2/23

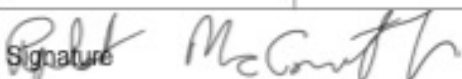
### Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).


Title of Paper	Lifting 2D Object Locations to 3D by Discounting LiDAR Outliers across Objects and Views
Publication Status	<input type="checkbox"/> Published
Publication Details	McCraith, R., Insafutdinov, E., Neumann, L., Vedaldi, A.  ICRA 2022

#### Student Confirmation

Student Name:	Robert McCraith		
Contribution to the Paper	performed experiments, wrote code, interpreted data, wrote manuscript.		
Signature 	Date	11/2/23	

#### Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Andrea Vedaldi		
Supervisor comments  This is an accurate description of the contributions.		
Signature 	Date	11/2/23

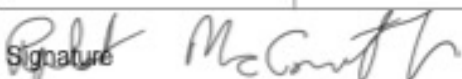
### Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).


Title of Paper	Direct LiDAR-based object detector training from automated 2D detections
Publication Status	<input type="checkbox"/> Published
Publication Details	McCraith, R., Insafutdinov, E., Neumann, L., Vedaldi, A.  NeurIPS 2022 Workshop on Machine Learning for Autonomous Driving

#### Student Confirmation

Student Name:	Robert McCraith		
Contribution to the Paper	performed experiments, wrote code, interpreted data, wrote manuscript.		
Signature 	Date	11/2/23	

#### Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Andrea Vedaldi		
Supervisor comments  This is an accurate description of the contributions.		
Signature 	Date	11/2/23



# Bibliography

- [1] CVPR 2017 workshop on autonomous driving. <https://github.com/TuSimple/tusimple-benchmark>. Accessed: 2020-02-15.
- [2] Parallel domain: Data generation for autonomy. <https://www.paralleldomain.com/>.
- [3] Mario Bijelic, Tobias Gruber, and Werner Ritter. A benchmark for lidar sensors in fog: Is detection breaking down? In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 760–767. IEEE, 2018.
- [4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nusenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.
- [5] Benjamin Caine, Rebecca Roelofs, Vijay Vasudevan, Jiquan Ngiam, Yuning Chai, Z. Chen, and Jonathon Shlens. Pseudo-labeling for scalable 3d object detection. *ArXiv*, abs/2103.02093, 2021.
- [6] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *CoRR*, abs/2005.12872, 2020.
- [7] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.
- [8] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7063–7072, 2019.

- [9] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [10] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.
- [11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.
- [12] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.
- [13] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, pages 2650–2658, 2015.
- [14] Nolang Fanani, Matthias Ochs, Alina Sturck, and Rudolf Mester. CNN-based multi-frame IMO detection from a monocular camera. In *2018 IEEE Intelligent Vehicles Symposium, IV 2018, Changshu, Suzhou, China, June 26-30, 2018*, pages 957–964, 2018.
- [15] Di Feng, Lars Rosenbaum, Fabian Timm, and Klaus Dietmayer. Labels Are Not Perfect: Improving Probabilistic Object Detection via Label Uncertainty. *arXiv e-prints*, page arXiv:2008.04168, August 2020.
- [16] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, June 2018.
- [17] A Gaidon, Q Wang, Y Cabon, and E Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016.
- [18] A Gaidon, Q Wang, Y Cabon, and E Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016.

- [19] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [20] Clément Godard, Oisín Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.
- [21] Clément Godard, Oisín Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *CVPR*, pages 3828–3838, 2019.
- [22] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [23] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.
- [24] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. *CoRR*, abs/1612.01925, 2016.
- [25] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *2010 20th International Conference on Pattern Recognition*, pages 2756–2759, Aug 2010.
- [26] Moritz Kampelmühler, Michael G Müller, and Christoph Feichtenhofer. Camera-based vehicle velocity estimation from monocular video. *arXiv preprint arXiv:1802.07094*, 2018.
- [27] L. Koestler, N. Yang, R. Wang, and D. Cremers. Learning monocular 3d vehicle detection without 3d bounding box labels. In *Proceedings of the German Conference on Pattern Recognition (GCPR)*, 2020.
- [28] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.

- [29] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.
- [30] Qinghao Meng, Wenguan Wang, Tianfei Zhou, Jianbing Shen, Luc Van Gool, and Dengxin Dai. Weakly supervised 3D object detection from LiDAR point cloud. In *ECCV*, 2020.
- [31] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgb-d images. In *ECCV*, 2012.
- [32] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. *arXiv preprint arXiv:1711.08488*, 2017.
- [33] Zengyi Qin, Jinglu Wang, and Yan Lu. Weakly supervised 3D object detection from point clouds. In *Proc. ACMM*, 2020.
- [34] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.
- [35] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [36] Brandon Wagstaff and Jonathan Kelly. Self-supervised scale recovery for monocular depth and egomotion estimation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2620–2627, 2021.
- [37] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

- [38] Junyuan Xie, Ross Girshick, and Ali Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *ECCV*, pages 842–857. Springer, 2016.
- [39] Koichiro Yamaguchi, Takeo Kato, and Yoshiki Ninomiya. Vehicle ego-motion estimation and moving object detection using a monocular camera. In *18th International Conference on Pattern Recognition (ICPR’06)*, pages 610–613. IEEE, 2006.
- [40] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *CVPR*, 2021.
- [41] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [42] Sergey Zakharov, Wadim Kehl, Arjun Bhargava, and Adrien Gaidon. Autolabeling 3D objects with differentiable rendering of SDF shape priors. In *Proc. CVPR*, 2020.
- [43] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, pages 1851–1858, 2017.
- [44] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.