OBJECT DETECTION FROM 2D TO 3D

By

Garrick Brazil

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science — Doctor of Philosophy

2021

ABSTRACT

OBJECT DETECTION FROM 2D TO 3D

By

Garrick Brazil

Monocular camera-based object detection plays a critical role in widespread applications including robotics, security, self-driving cars, augmented reality and many more. Increased relevancy is often given to the detection and tracking of safety-critical objects like pedestrians, cyclists, and cars which are often in motion and in close association to *people*. Compared to other generic objects such as animals, tools, food — safety-critical objects in urban scenes tend to have unique challenges. Firstly, such objects usually have a wide range of detection scales such that they may appear anywhere from $5 - 50+$ meters from the camera. Safety-critical objects also tend to have a high variety of textures and shapes, exemplified by the clothing of people and variability of vehicle models. Moreover, the high-density of objects in urban scenes leads to increased levels of self-occlusion compared to general objects in the wild. Off-the-shelf object detectors do not always work effectively due to these traits, and hence special attention is needed for accurate detection. Moreover, even successful detection of safety-critical is not *inherently* practical for applications designed to function in the real 3D world, without integration of expensive depth sensors.

To remedy this, in this thesis we aim to improve the performance of 2D object detection and extend boxes into 3D, while using **only** monocular camera-based sensors. We first explore how pedestrian detection can be augmented using an efficient simultaneous detection and segmentation technique, while notably requiring no additional data or annotations. We then propose a multi-phased autoregressive network which progressively improves pedestrian detection precision for difficult samples, while critically maintaining an efficient runtime. We additionally propose a

single-stage region proposal networks for 3D object detection in urban scenes, which is both more efficient and up to $3\times$ more accurate than comparable state-of-the-art methods. We stabilize our 3D object detector using a highly tailored 3D Kalman filter, which both improves localization accuracy and provides useful byproducts such as ego-motion and per-object velocity. Lastly, we utilize differentiable rendering to discover the underlying 3D structure of objects beyond the cuboids used in detection, and without relying on expensive sensors or 3D supervision. For each method, we provide comprehensive experiments to demonstrate effectiveness, impact and runtime efficiency.

This thesis is dedicated to my loving family and in particular to my late aunt Cynthia, who believed in me and inspires me always.

ACKNOWLEDGMENTS

The journey through graduate school and leading to this dissertation would not have been possible without the help of many mentors, friends, and family.

Firstly, I would like to thank my adviser Dr. Xiaoming Liu, whose encouragement and mentorship has led me to learn and accomplish goals I never anticipated possible for myself. I am forever grateful to have been given the chance to learn and grow in the lab with invaluable guidance. His knowledge, experience, and enthusiasm for research are a continued inspiration for me. As result I feel I have grown as a person in critical thinking, writing, presenting, and conducting research as a whole. Besides academics, Dr. Liu's unceasing support and advice during difficult times were not only valued, but were paramount to my well-being and making it this far ... I am truly thankful for the patience and counsel in those times especially.

I would like to express my gratitude to my committee members Dr. Arun Ross, Dr. Vishnu Boddeti and Dr. Daniel Morris for their invaluable insight, advice, and mentorship — pertaining to the development of this thesis and in countless other meetings throughout my time at MSU.

I would further like to thank Dr. Gerard Pons-Moll and Dr. Bernt Schiele for their mentorship during my visit abroad in Saarbrücken, Germany and afterwards. Their advice and feedback were vital to my growth as a researcher. Moreover, the warm welcome into the MPI group is a time in my life I will always reflect back fondly on.

I am thankful to my mentors Georgia Gkioxari, Justin Johnson and Nikhila Ravi for their help, patience, and substantial counseling during my internship. I am grateful for everything I learned as a researcher and person, while they helped me climb a rather intimidating learning curve. I cherish the kindness I was shown from all conceivable directions from the entire FAIR group. Despite coinciding with a turbulent year — 2020 — the mentorship and research experience resulted

in one of the most delightful and intriguing periods of my professional career.

It is my pleasure to be labmates with a wonderful and smart group of individuals. I would like to thank all my labmates (past and present) Dr. Joseph Roth, Dr. Xi Yin, Dr. Amin Jourabloo, Dr. Morteza Safdarnejad, Dr. Yousef Atoum, Dr. Luan Tran, Dr. Feng Liu, Yaojie Liu, Adam Terwilliger, Joel Stehouwer, Bangjie Yin, Hieu Nguyen, Shengjie Zhu, Masa Hu, Andrew Hou, Abhinav Kumar, Ying Tai, Vishal Asnani, Xiao Guo, and Ziyuan Zhang. Thank you for fostering a kind working atmosphere. I appreciate the help, perspectives, impromptu discussions, and collaborations. The memories we have formed together in board game nights, movies, hiking, travels, *etc*., have made my PhD journey far more enjoyable, enriching, and . . . entertaining too.

Finally, I would like to thank **all** of my family and friends. In particular, to my loving mother Colleen, my cousin Richard and my uncle Werner — thank you for the support and care through an emotional and rather stressful peak of my life. I couldn't have made it without your help. To my support circle and close friends Andrew, Adam, Kristen, Laura, Kushal and Stella — thank you for the kind support, delightful memories, and helping me keep a healthy brain. To my canine friends, Sylar and Raelynn, thank you for your scientific advice and interesting insights.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

Chapter 1

Introduction and Contributions

1.1 Introduction

Object detection is an important component which many applications depend on. Examples of such applications include robotics [42], security systems [85], and urban autonomous driving [51]. The task of object detection involves *both* the classification and localization of all objects in a visible scene whether they be in a 2D image space in pixels or 3D camera space in meters. Compared to other domains, the domain of urban autonomous driving scenarios tends to deal primarily with safety-critical classes including pedestrians, cyclists, and cars. The ability to reliably detect such objects is important to enabling the future of autonomous vehicles (AV). As such, the power and robustness of object detection and has made significant progress in recent years [13, 98, 99, 130].

However, many unique challenges remain unsolved for urban objects compared to that of generic objects. A notable difference is in the range of scales of the relevant objects. It is common for objects, even among the same category, to move in a dynamic range between $5 - 50+$ meters away from the capturing camera [51]. Therefore, the appearance of said objects have high intraclass variation and at far ranges become highly blurred and non-discriminative. The shape and

textures further tends to have very high variability compared to objects generically. This effect is exemplified by the variation of clothing on pedestrians and in the diverse shapes of vehicle models. Additionally, urban self-driving scenarios commonly involve significant dynamic motion which makes modeling in video more troublesome. For instance, such motion is derived from both the foreground objects and the capturing camera, typically mounted on the roof of a *moving* vehicle.

Prevalent methods for monocular object detection have focused primarily on object detection of 2D bounding boxes [76, 166, 184, 188]. We emphasize that although 2D bounding boxes may be beneficial for other applications (security, face, and generic object detection), within the AV domain it is *critical* to localize objects in 3D space for path planning and collision avoidance.

Many 3D sensors are available to aid in urban object detection. Notable sensor configurations for object detection include cameras, LiDAR, and RADAR. Cameras typically have the lowest cost and the highest availability compared the other aforementioned sensors. They uniquely capture both the shape *and* texture information of a scene, the latter of which is not present in other sensors [16]. However, a critical drawback of camera systems is in its inherent ambiguity in depth, which LiDAR and RADAR comparably excel at. Multi-camera systems partially address such limitations by using a known 3D geometry and calibration to approximate depth more accurately. We opt to study the problem of object detection using a single-view *monocular camera* due to its low cost and accessibility compared to all other configurations.

In this dissertation, we study and present methods to solve both monocular 2D and 3D object detection with emphasis on safety-critical objects in urban autonomous driving scenarios. Starting from seminal work on Faster R-CNN [132], we analyze the effects of incorporating additional critical supervision signals and its effects on object detection accuracy. Specifically, we approach the problem from the perspective of incorporating weak shape supervision, intermediate network supervision, 3D bounding box supervision, video cues, and 3D mesh discovery. Despite studying

the effects in isolation, each novel technique is compatible to a high-level base framework and presented as modular extensions of a theoretical shared system.

We begin with a novel method to utilize shape supervision via weak segmentation applied to the task of pedestrian detection. We identify the similarities of segmentation and detection and demonstrate how their joint supervision using weak box-based segmentation can greatly improve the localization of pedestrians. We find that the robustness of segmentation shape helps correct the detection of unusually shaped pedestrians (sitting, bent over, or occluded), which dominate the corrections. We emphasize that by using weak box-based segmentation masks no additional data is required. Further, since the segmentation is used only to improve training, there is no added overhead to the method at inference. Overall, our method improves the state-of-the-art on competitive datasets [38, 51] and executes $\approx 2\times$ faster than competitive methods of the time.

We then expand our framework to include intermediate network supervision designed to iteratively improve the precision of object detection. Specifically, we build autoregressive phases *into* a region proposal network which is optimized with intermediate supervision of increasingly difficult labels. We emphasize that the phases are built *within* the network such that more advanced features can be learned via an efficient lightweight decoder-encoder structure. A competitive network runtime efficiency is maintained since since the phases are designed inside of the network rather than adding additional separate networks as an ensemble. We provide detailed analysis to demonstrate how each phase becomes more precise and produces higher peaks at the centroid of unique pedestrians. Collectively, our autoregressive method strikes a favorable balance between accuracy and runtime, while improving the state-of-the-art for pedestrian detection even in notably difficult high-occlusion evaluation settings.

We next incorporate 3D supervision into the general 2D region proposal network. We note that perceiving objects in 3D is inherently more valuable for real-world applications and particularly so

in the domain of AV. Compared to our technique, prior works on monocular 3D object detection utilize multiple separate networks, stages, datasets, and are typically not trained end-to-end. In contrast, we propose a single 3D region proposal network which takes a monocular image as input and is trained fully end-to-end. We further utilize the relation of 2D and 3D geometry in fixed camera scenarios by using a novel depth-aware convolutional operator. Our method significantly improves both Bird's Eye View and 3D object detection accuracy ($\approx 3\times$) despite using only a single region proposal network. Moreover, our proposed approach is more future-proof since its core concepts can be naturally integrated into any monocular 3D object detection system which uses a 2D region proposal network, resulting in a 3D proposal which *already* performs well.

We note that although motion is important for human vision in detection, tracking and perceiving depth, it is not thoroughly utilized in monocular 3D object detection systems. We next extend our system to leverage monocular video streams using a novel 3D Kalman filter. Collectively, the integration of video helps stabilize noisy object localizations in the 3D world. We observe our kinematic method results in an appreciable improvement to localization accuracy and produces useful byproducts such as ego-motion and per-object velocity.

We have discussed detecting and localizing objects from 2D to 3D, but only coarsely in the representation of 3D cuboids. We next explore how to perceive such objects more finely as 3D meshes. We emphasize that 3D mesh ground truths are *expensive* to annotate and are generally not available for in the wild scenarios and datasets. We therefore propose a general end-to-end approach to discover 3D mesh shapes from only a collection of unassociated images and 2D mask supervision — a common and cost-effective label. We accomplish this goal by jointly learning a template mesh, object pose, and refined shapes from only 2D labels. We scale our method to a diverse set of in the wild object categories beyond cars due to the freedom from 3D supervision.

In summary, we provide novel solutions to localize and understand objects aided by unique

techniques and signals including weak segmentation supervision, intermediate supervision, 3D supervision, temporal cues, and lastly mesh signals. Our methods pay special attention to safety-critical objects including pedestrians, cyclists, and cars — commonly found in urban landscapes. Notably, each method is tailored to maintain or improve the runtime efficiency, while appreciably improving accuracy relative to comparable state-of-the-art methods. We emphasize that our methods help improve **object detection from 2D to 3D**, and thereby drastically increase the effective utility in practice — for as long as we live in a 3D world.

1.2 Dissertation Contributions

This dissertation studies the problem of object detection ranging from 2D detection of pedestrians to multi-class 3D object detection with additional novelties in both video and 3D mesh discovery. We summarize the collective contributions of this dissertation as follows:

- We propose a multi-task network which enhances monocular pedestrian detection using simultaneous detection and segmentation. We avoid requiring any additional data or inference time by only re-using the bounding box annotations as weak segmentation labels. We provide insights on the unique characteristics of weak segmentation compared to typical object detection paradigms, and its illuminating effects on convolutional feature maps.
- We simultaneously address the unique scale challenges suffered in prior pedestrian detection work [183] when adapting Faster R-CNN [132] by a novel *fusion* of predictions from a region proposal network and a binary classifier when trained with a stricter supervision signal.
- We further improve our framework by developing a multi-phased autoregressive pedestrian detection system built *into* a region proposal network. Notably, each phase of our network is trained on progressively more strict supervision signals which iteratively improve precision.

- We propose a lightweight decoder-encoder module within the autoregressive pedestrian detection system to enable both feature map refinement and message passing between phases while using convolutional re-sampling for efficient feature map pathways.
- We formulate a novel 3D object detector using a monocular 3D regional proposal network (M3D-RPN) which operates in a shared 2D and 3D object detection space. Prior statistics are precomputed as anchors used for strong initialization, which help ease the regression of 3D parameters. Our method utilizes a single network and is trained end-to-end, compared to prior art [23, 24, 118, 172] which require external networks, data, and representations.
- We propose a novel operation for depth-aware convolution in order to improve estimation of 3D parameters. In effect, the network is able to learn high-level and spatially aware features, which are able to vary for objects which are *close* or *far* from the camera.
- We propose video-based approach for monocular 3D object detection which leverages realistic motion constraints using an integrated ego-motion estimator and a 3D Kalman filter.
- Our video-based system uses only a single model to more comprehensively understand the 3D dynamics of a scene, resulting in more accurate 3D localization and useful byproducts including ego-motion and per-object velocity.
- We form a novel method which goes beyond simple 3D cuboids to *discover* 3D mesh representation from unassociated images using only 2D mask supervision, which we comprehensively evaluate on both cars and $20+$ additional in the wild categories.
- Our 3D mesh discovery framework is able to jointly learn a shared template shape, object pose and refined instance-level meshes in an end-to-end manner, powered by a novel template learner, camera multiplexer and cluster consistency constraint.

1.3 Dissertation Organization

We organize the remaining chapters of the dissertation as follows. Chapter 2 proposes the novel framework for simultaneous detection and segmentation as an augment to pedestrian detection. Chapter 3 develops a framework for autoregressive multi-phased pedestrian detection built into a region proposal network. Chapter 4 explores how 3D object detection can perform accurately with only a single region proposal network as well as the proposed depth-aware convolution. Chapter 5 extends our 3D object detection system to work with video using a tailored 3D Kalman filter for both better stabilization and a more comprehensive understanding of scene dynamics. Chapter 6 explores our method for discovering objects in 3D mesh representations rather than coarse cuboids, while only utilizing 2D signals for supervision. Chapter 7 concludes the dissertation.

Chapter 2

Illuminating Pedestrians via Simultaneous Detection & Segmentation

Pedestrian detection is a critical problem in computer vision with significant impact on safety in urban autonomous driving. In this work, we explore how semantic segmentation can be used to boost pedestrian detection accuracy while having little to no impact on network efficiency. We propose a segmentation infusion network to enable joint supervision on semantic segmentation and pedestrian detection. When placed properly, the additional supervision helps guide features in shared layers to become more sophisticated and helpful for the downstream pedestrian detector. Using this approach, we find weakly annotated boxes to be sufficient for considerable performance gains. We provide an in-depth analysis to demonstrate how shared layers are shaped by the segmentation supervision. In doing so, we show that the resulting feature maps become more semantically meaningful and robust to shape and occlusion. Overall, our simultaneous detection and segmentation framework achieves a considerable gain over the state-of-the-art on the Caltech pedestrian dataset, competitive performance on KITTI, and executes $2\times$ faster than competitive methods.

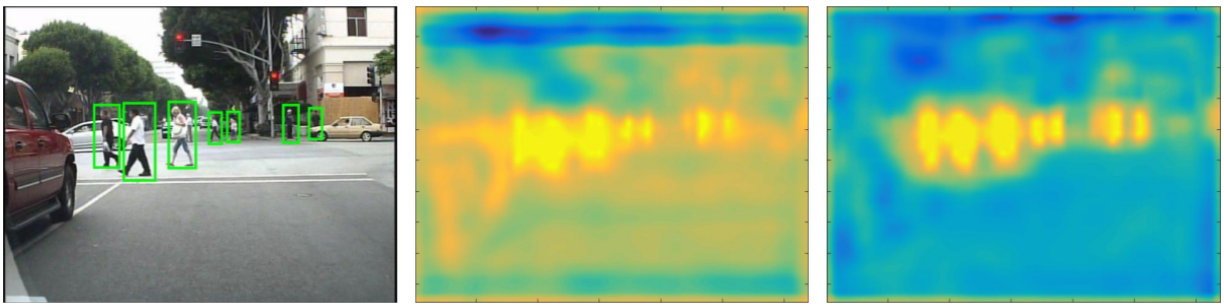

Figure 2.1: Detection results on the Caltech test set (left), feature map visualization from the RPN of conventional Faster R-CNN (middle), and feature map visualization of SDS-RCNN (right). Notice that our feature map substantially illuminates the pedestrian shape while suppressing the background region, both of which make positive impact to downstream pedestrian detection.

2.1 Introduction

Pedestrian detection from an image is a core capability of computer vision, due to its applications such as autonomous driving and robotics [51]. It is also a long-standing vision problem because of its distinct challenges including low resolution, occlusion, cloth variations, etc [184]. There are two central approaches for detecting pedestrians: object detection [13, 183] and semantic segmentation [21, 31]. The two approaches are highly related by nature but have their own strengths and weaknesses. For instance, object detection is designed to perform well at localizing distinct objects but typically provides little information on object boundaries. In contrast, semantic segmentation does well at distinguishing pixel-wise boundaries among classes but struggles to separate objects within the same class.

Intuitively, we expect that knowledge from either task will make the other substantially easier. This has been demonstrated for generic object detection, since having segmentation masks of objects would clearly facilitate detection. For example, Fidler et al. [47] utilize predicted segmentation masks to boost object detection performance via a deformable part-based model. Hariharan et al. [62] show how segmentation masks generated from MCG [2] can be used to mask background

regions and thus simplify detection. Dai et al. [32] utilize the two tasks in a 3-stage cascaded network consisting of box regression, foreground segmentation, and classification. Their architecture allows each task to share features and feed into one another.

In contrast, the pairing of these two tasks is rarely studied in pedestrian detection, despite the recent advances [13, 91, 183]. This is due in part to the lack of pixel-wise annotations available in classic pedestrian datasets such as Caltech [38] and KITTI [51], unlike the detailed segmentation labels in the COCO [100] dataset for generic object detection. With the release of Cityscapes [31], a high quality dataset for urban semantic segmentation, it is expected that substantial research efforts will be on *how to leverage semantic segmentation to boost the performance of pedestrian detection*, which is the core problem to be studied in this paper.

Given this objective, we start by presenting a competitive two-stage baseline framework of pedestrian detection deriving from RPN+BF [183] and Faster R-CNN [132]. We contribute a number of key changes to enable the second-stage classifier to specialize in stricter supervision and additionally fuse the refined scores with the first stage RPN. These changes alone lead to state-of-the-art performance on the Caltech benchmark. We further present a simple, but surprisingly powerful, scheme to utilize multi-task learning on pedestrian detection and semantic segmentation. Specifically, we infuse the semantic segmentation mask into shared layers using a *segmentation infusion layer* in both stages of our network. We term our approach as "simultaneous detection and segmentation R-CNN (SDS-RCNN)". We provide an in-depth analysis on the effects of joint training by examining the shared feature maps, e.g., Fig. 2.1. Through infusion, the shared feature maps begin to illuminate pedestrian regions. Further, since we infuse the semantic features during training only, the network efficiency at inference is unaffected. We demonstrate the effectiveness of SDS-RCNN by reporting considerable improvement (23% relative reduction of the error) over the published state-of-the-art on Caltech [38], competitive performance on KITTI [51], and a runtime

roughly $2\times$ faster than competitive methods.

In summary our contributions are as follows:

- Improved baseline derived from [132, 183] by enforcing stricter supervision in the second-stage classification network, and further fusing scores between stages.
- A multi-task infusion framework for joint supervision on pedestrian detection and semantic segmentation, with the goal of *illuminating* pedestrians in shared feature maps and easing downstream classification.
- We achieve the new state-of-the-art performance on Caltech pedestrian dataset, competitive performance on KITTI, and obtain $2\times$ faster runtime.

2.2 Prior work

Object Detection: Deep convolution neural networks have had extensive success in the domain of object detection. Notably, derivations of Fast [54] and Faster R-CNN [132] are widely used in both generic object detection [13, 53, 175] and pedestrian detection [91, 152, 183]. Faster R-CNN consists of two key components: a region proposal network (RPN) and a classification sub-network. The RPN works as a sliding window detector by determining the *objectness* across a set of predefined anchors (box shapes defined by aspect ratio and scale) at each spatial location of an image. After object proposals are generated, the second stage classifier determines the precise class each object belongs to. Faster R-CNN has been shown to reach state-of-the-art performance on the PASCAL VOC 2012 [44] dataset for generic object detection and continues to serve as a frequent baseline framework for a variety of related problems [53, 62, 65, 184].

Pedestrian Detection: Pedestrian detection is one of the most extensively studied problems in

object detection due to its real-world significance. The most notable challenges are caused by small scale, pose variations, cyclists, and occlusion [184]. For instance, in the Caltech pedestrian dataset [38] 70% of pedestrians are occluded in at least one frame.

The top performing approaches on the Caltech pedestrian benchmark are variations of Fast or Faster R-CNN. SA-FastRCNN [54] and MS-CNN [13] reach competitive performance by directly addressing the scale problem using specialized multi-scale networks integrated into Fast and Faster R-CNN respectively. Furthermore, RPN+BF [183] shows that the RPN of Faster R-CNN performs well as a standalone detector while the downstream classifier degrades performance due to collapsing bins of small-scale pedestrians. By using higher resolution features and replacing the downstream classifier with a boosted forest, RPN+BF is able to alleviate the problem and achieve 9.58% miss rate on the Caltech reasonable [39] setting. F-DNN [41] also uses a derivation of the Faster R-CNN framework. Rather then using a single downstream classifier, F-DNN fuses multiple parallel classifiers including ResNet [65] and GoogLeNet [150] using soft-reject and further incorporates multiple training datasets to achieve 8.65% miss rate on the Caltech reasonable setting. The majority of top performing approaches utilize some form of a RPN, whose scores are typically discarded after selecting the proposals. In contrast, our work shows that fusing the score with the second stage network can lead to substantial performance improvement.

Simultaneous Detection & Segmentation: There are two lines of research on simultaneous detection and segmentation. The first aims to improve the performance of both tasks, and formulates a problem commonly known as *instance-aware semantic segmentation* [31]. Hariharan et al. [62] predict segmentation masks using MCG [2] then get object instances using “slow" R-CNN [55] on masked image proposals. Dai et al. [32] achieve high performance on instance segmentation using an extension of Faster R-CNN in a 3-stage cascaded network including mask supervision.

The second aims to explicitly improve object detection by using segmentation as a strong cue.

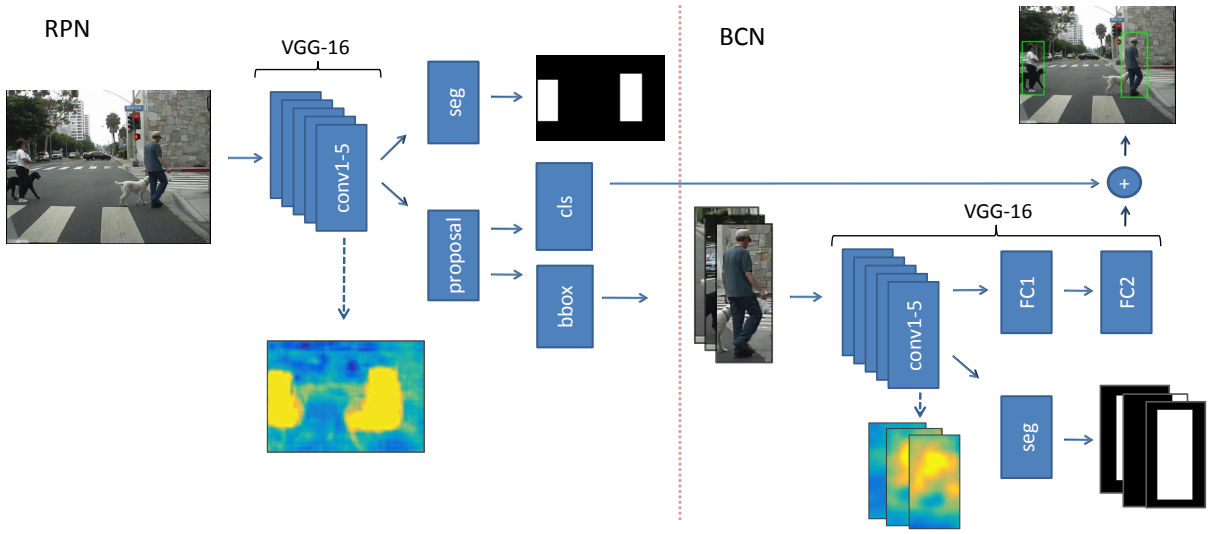

Figure 2.2: Overview of the proposed SDS-RCNN framework. The segmentation layer infuses semantic features into shared conv1-5 layers of each stage, thus *illuminating* pedestrians and easing downstream pedestrian detection (proposal layers in RPN, and FC1-2 in BCN).

Early work on the topic by Fidler et al. [47] demonstrates how semantic segmentation masks can be used to extract strong features for improved object detection via a deformable part-based model. Du et al. [41] use segmentation as a strong cue in their F-DNN+SS framework. Given the segmentation mask predicted by a third parallel network, their ensemble network uses the mask in a post-processing manner to suppress background proposals, and pushes performance on the Caltech pedestrian dataset from 8.65% to 8.18% miss rate. However, the segmentation network degrades the efficiency of F-DNN+SS from 0.30 to 2.48 seconds per image, and requires multiple GPUs at inference. In contrast, our novel framework infuses the semantic segmentation masks into shared feature maps and thus does not require a separate segmentation network, which outperforms [41] in both accuracy and network efficiency. Furthermore, our use of weak box-based segmentation masks addresses the issue of lacking pixel-wise segmentation annotations in [38, 51].

2.3 Proposed method

Our proposed architecture consists of two key stages: a region proposal network (RPN) to generate candidate bounding boxes and corresponding scores, and a binary classification network (BCN) to refine their scores. In both stages, we propose a *semantic segmentation infusion layer* with the objective of making downstream classification a substantially easier task. The infusion layer aims to encode semantic masks into shared feature maps which naturally serve as strong cues for pedestrian classification. Due to the impressive performance of the RPN as a standalone detector, we elect to fuse the scores between stages rather than discarding them as done in prior work [13, 41, 154, 183]. An overview of the SDS-RCNN framework is depicted in Fig. 2.2

2.3.1 Region Proposal Network

The RPN aims to propose a set of bounding boxes with associated confidence scores around potential pedestrians. We adopt the RPN of Faster R-CNN [132] following the settings in [183]. We tailor the RPN for pedestrain detection by configuring $N_a = 9$ anchors with a fixed aspect ratio of 0.41 and spanning a scale range from 25 – 350 pixels, corresponding to the pedestrain statistics of Caltech [38]. Since each anchor box acts as a sliding window detector across a pooled image space, there are $N_p = N_a \times \frac{W}{fs} \times \frac{H}{fs}$ total pedestrian proposals, where f_s corresponds to the feature stride of the network. Hence, each proposal box i corresponds to an anchor and a spatial location of image $\mathbf{I}$.

The RPN architecture uses conv1-5 from VGG-16 [141] as the backbone. Following [132], we attach a proposal feature extraction layer to the end of the network with two sibling output layers for box classification (*cls*) and bounding box regression (*bbox*). We further add a segmentation infusion layer to conv5 as detailed in Sec. 2.3.3.

For every proposal box i, the RPN aims to minimize the following joint loss function with three terms:

$$L = \lambda_c \sum_i L_c(c_i, \hat{c_i}) + \lambda_r \sum_i L_r(t_i, \hat{t_i}) + \lambda_s L_s. \tag{2.1}$$

The first term is the classification loss L_c, which is a softmax logistic loss over two classes (pedestrian vs. background). We use the standard labeling policy which considers a proposal box at location i to be pedestrian ($c_i = 1$) if it has at least 0.5 Intersection over Union (IoU) with a ground truth pedestrian box, and otherwise background ($c_i = 0$). The second term seeks to improve localization via bounding box regression, which learns a transformation for each proposal box to the nearest pedestrian ground truth. Specifically, we use $L_r(t_i, \hat{t_i}) = R(t_i - \hat{t_i})$ where R is the robust (smooth L_1) loss defined in [54]. The bounding box transformation is defined as a 4-tuple consisting of shifts in x, y and scales in w, h denoted as $t = [t_x, t_y, t_w, t_h]$. The third term L_s is the segmentation loss presented in Sec. 2.3.3.

In order to reduce multiple detections of the same pedestrian, we apply non-maximum suppression (NMS) greedily to all pairs of proposals after the transformations have been applied. We use an IoU threshold of 0.5 for NMS.

We train the RPN in the Caffe [71] framework using SGD with a learning rate of 0.001, momentum of 0.9, and mini-batch of 1 full-image. During training, we randomly sample 120 proposals per image at a ratio of $1{:}5$ for pedestrian and background proposals to help alleviate the class imbalance. All other proposals are treated as ignore. We initialize conv1-5 from a VGG-16 model pretrained on ImageNet [36], and all remaining layers randomly. Our network has four max-pooling layers (within conv1-5), hence $f_s = 16$. In our experiments, we regularize our multi-task loss terms by setting $\lambda_c = \lambda_s = 1, \lambda_r = 5$.

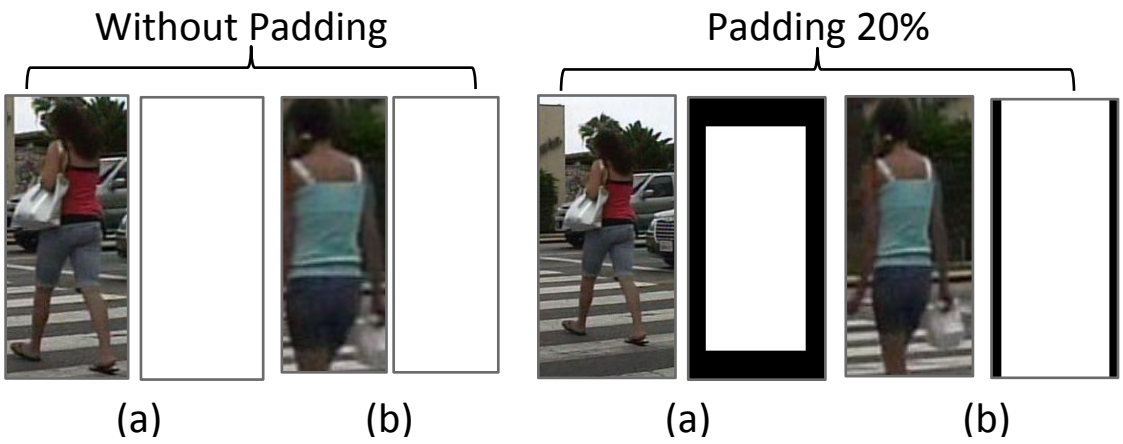

Figure 2.3: Example proposal masks with and without padding. There is no discernible difference between the non-padded masks of well-localized (a) and poorly localized (b) proposals.

2.3.2 Binary Classification Network

The BCN aims to perform pedestrian classification over the proposals of the RPN. For generic object detection, the BCN usually uses the downstream classifier of Faster R-CNN by sharing conv1-5 with the RPN, but was shown by [183] to degrade pedestrian detection accuracy. Thus, we choose to construct a separate network using VGG-16. The primary advantage of a separate network is to allow the BCN freedom to specialize in the types of “harder" samples left over from the RPN. While sharing computation is highly desirable for the sake of efficiency, the shared networks are more predestined to predict similar scores which are redundant when fused. Therefore, rather than cropping and warping a shared feature space, our BCN directly crops the top N_b proposals from the RGB input image.

For each proposal image i, the BCN aims to minimize the following joint loss function with two terms:

$$L = \lambda_c \sum_i w_i L_c(c_i, \hat{c_i}) + \lambda_s L_s. \tag{2.2}$$

Similar to RPN, the first term is the classification loss L_c where c_i is the class label for the ith

proposal. A cost-sensitive weight w_i is used to give precedence to detect large pedestrians over small pedestrians. There are two key motivations for this weighting policy. First, large pedestrians typically imply close proximity and are thus significantly more important to detect. Secondly, we presume that features of large pedestrians may be more helpful for detecting small pedestrians. We define the weighting function given the ith proposal with height h_i and a pre-computed mean height $\bar{h}$ as $w_i = 1 + \frac{h_i}{\bar{h}}$. The second term is the segmentation loss presented in Sec. 2.3.3.

We make a number of significant contributions to the BCN. First, we change the labeling policy to encourage higher precision and further diversification from the RPN. We enforce a *stricter* labeling policy, requiring a proposal to have IoU > 0.7 with a ground truth pedestrian box to be considered pedestrian ($c_i = 1$), and otherwise background ($c_i = 0$). This encourages the network to suppress poorly localized proposals and reduces false positives in the form of double detections. Secondly, we choose to fuse the scores of the BCN with the confidence scores of the RPN at test time. Since our design explicitly encourages the two stages to diversify, we expect the classification characteristics of each network to be complementary when fused. We fuse the scores at the feature level prior to softmax. Formally, the fused score for the ith proposal, given the predicted 2-class scores from the RPN $r_i = \{\hat{c}_i^0, \hat{c}_i^1\}$ and BCN $b_i = \{\hat{c}_i^0, \hat{c}_i^1\}$ is computed via a fused softmax function:

$$\hat{c_i} = \frac{e^{(\hat{r}_i^1 + \hat{b}_i^1)}}{e^{(\hat{r}_i^1 + \hat{b}_i^1)} + e^{(\hat{r}_i^0 + \hat{b}_i^0)}}, \tag{2.3}$$

where the superscript denotes the class corresponding channel ($0 =$ background, $1 =$ pedestrian). In effect, the fused scores become more confident when the stages agree, and otherwise lean towards the dominant score. Thus, it is ideal for each network to diversify in its classification capabilities such that at least one network may be *very* confident for each proposal.

For a modest improvement to efficiency, we remove the pool5 layer from the VGG-16 architecture then adjust the input size to 112×112 to keep the fully-connected layers intact. This is a fair trade-off since most pedestrian heights fall in the range of $30 - 80$ pixels [38]. Hence, small pedestrian proposals are upscaled by a factor of $\sim 2\times$, allowing space for finer discrimination. We further propose to pad each proposal by 20% on all sides to provide background context and avoid partial detections, as shown in Fig. 2.3.

We train the BCN in the Caffe [71] framework using the same settings as the RPN. We initialize conv1-5 from the trained RPN model, and all remaining layers randomly. During training, we set $N_b = 20$. During inference, we set $N_b = 15$ for a moderate improvement to efficiency. We regularize the multi-task loss by setting $\lambda_c = \lambda_s = 1$.

2.3.3 Simultaneous Detection & Segmentation

We approach simultaneous detection and segmentation with the motivation to make our downstream pedestrian detection task easier. We propose a segmentation infusion layer trained on weakly annotated pedestrian boxes which *illuminate* pedestrians in the shared feature maps preceding the classification layers. We integrate the infusion layer into both stages of our SDS-RCNN framework.

Segmentation Infusion Layer: The segmentation infusion layer aims to output two masks indicating the likelihood of residing on pedestrian or background segments. We choose to use only a single layer and a 1×1 kernel so the impact on the shared layers will be as high as possible. This forces the network to directly infuse semantic features into shared feature maps, as visualized in Fig. 2.4. A deeper network could achieve higher segmentation accuracy but will infer less from shared layers and diminish the overall impact on the downstream pedestrian classification. Further,

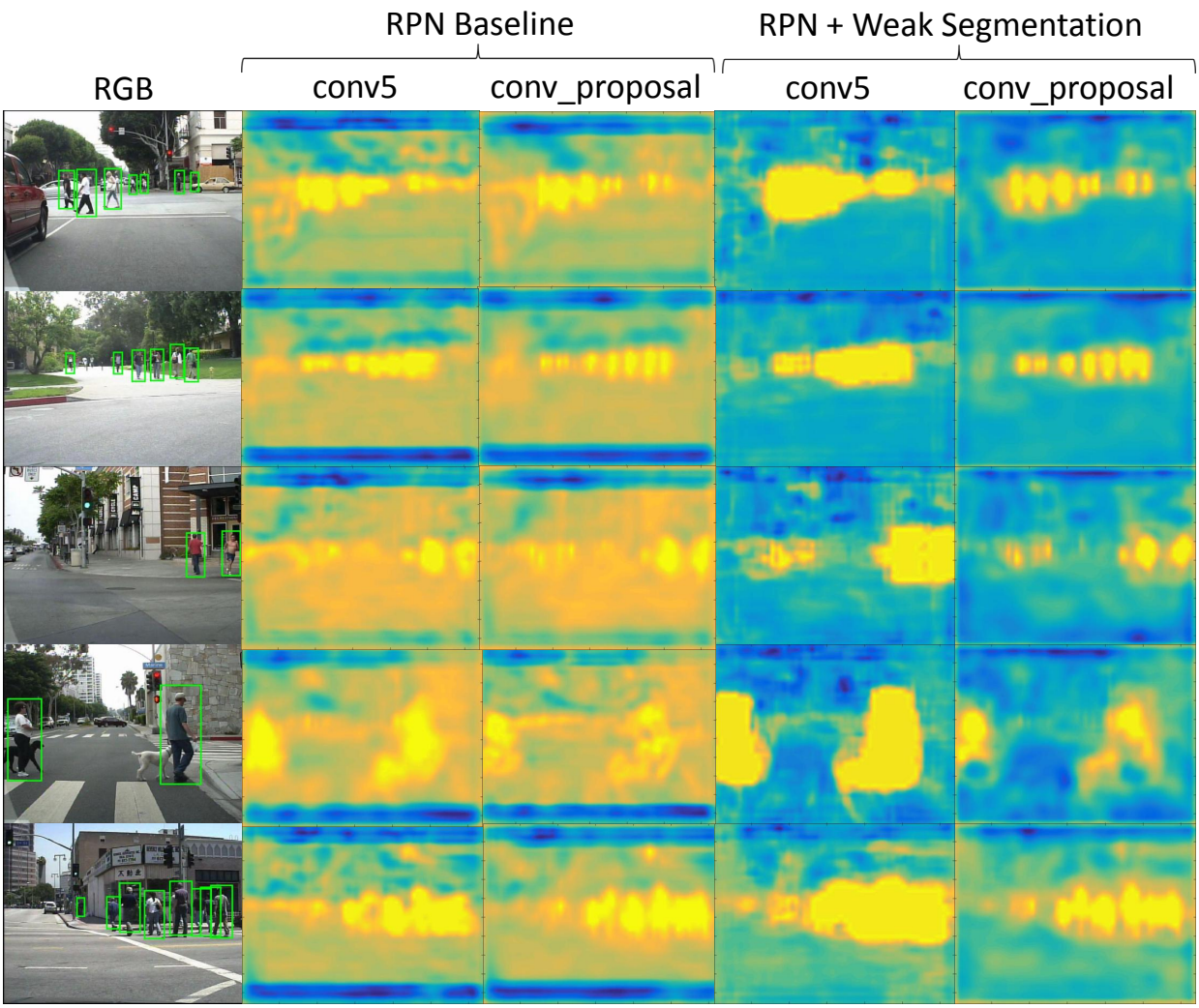

Figure 2.4: Feature map visualizations of conv5 and the proposal layer for the baseline RPN (left) and the RPN infused with weak segmentation supervision (right).

we choose to attach the infusion layer to conv5 since it is the deepest layer which precedes both the proposal layers of the RPN and the fully connected layers of the BCN.

Formally, the final loss term L_s of both the RPN and BCN is a softmax logistic loss over two classes (pedestrian vs. background), applied to each location i, where w_i is the cost-sensitive weight introduced in 2.3.2:

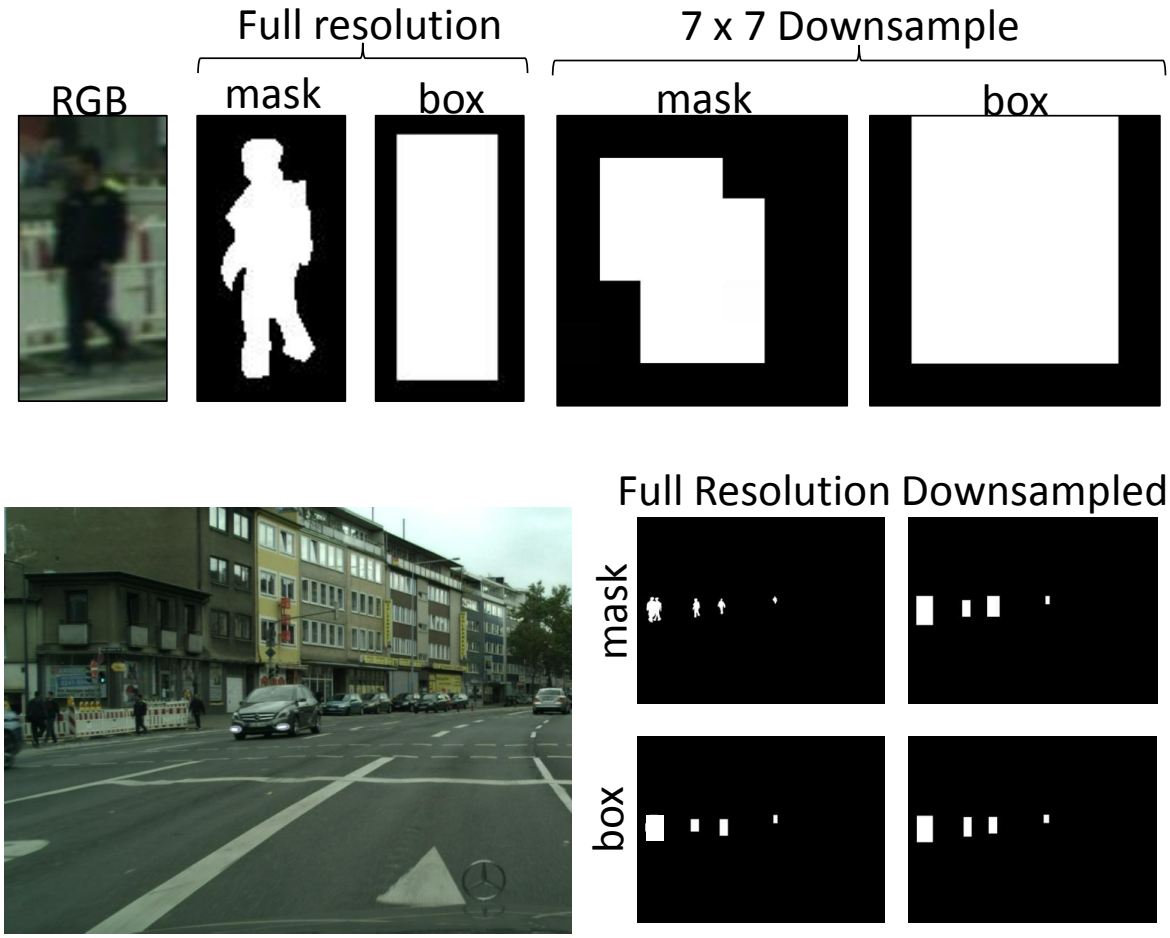

Figure 2.5: Visualization of the similarity between pixel-wise segmentation masks (from Cityscapes [31]) and weak box-based masks when downsampled in both the BCN (top) and RPN (bottom).

$$\lambda_s \sum_i w_i L_s(\mathbf{S}_i, \hat{\mathbf{S}}_i). \tag{2.4}$$

We choose to levereage the abundance of bounding box annotations available in popular pedestrian datasets (e.g., Caltech [38], KITTI [51]) by forming weak segmentation ground truth masks. Each mask $\mathbf{S} \in \mathbb{R}^{\mathbb{W} \times \mathbb{H}}$ is generated by labeling all pedestrian box regions as $\mathbf{S}_i = 1$, and otherwise background $\mathbf{S}_i = 0$. In most cases, box-based annotations would be considered too noisy for semantic segmentation. However, since we place the infusion layer at conv5, which has been

pooled significantly, the differences between box-based annotations and pixel-wise annotations diminish rapidly w.r.t. the pedestrian height (Fig. 2.5). For example, in the Caltech dataset 68% of pedestrians are less than 80 pixels tall, which corresponds to 3×5 pixels at conv5 of the RPN. Further, each of the BCN proposals are pooled to 7×7 at conv5. Hence, pixel-wise annotations may not offer a significant advantage over boxes at the high levels of pooling our networks undertake.

Benefits Over Detection: A significant advantage of segmentation supervision over detection is its simplicity. For detection, sensitive hyperparamters must be set, such as anchor selection and IoU thresholds used for labeling and NMS. If the chosen anchor scales are too sparse or the IoU threshold is too high, certain ground truths that fall near the midpoint of two anchors could be missed or receive low supervision. In contrast, semantic segmentation treats all ground truths indiscriminate of how well the pedestrian's shape or occlusion-level matches the chosen set of anchors. In theory, the incorporation of semantic segmentation infusion may help reduce the *sensitivity* of conv1-5 to such hyperparamters. Furthermore, the segmentation supervision is especially beneficial for the second stage BCN, which on its own would only know *if* a pedestrian is present. The infusion of semantic segmentation features inform the BCN *where* the pedestrian is, which is critical for differentiating poorly vs. well-localized proposals.

2.4 Experiments

We evaluate our proposed SDS-RCNN on popular datasets including Caltech [38] and KITTI [51]. We perform comprehensive analysis and ablation experiments using the Caltech dataset. We refer to our collective method as SDS-RCNN and our region proposal network as SDS-RPN. We show the performance curves compared to the state-of-the-art pedestrian detectors on Caltech in Fig. 2.6. We further report a comprehensive overview across datasets in Table 2.1.

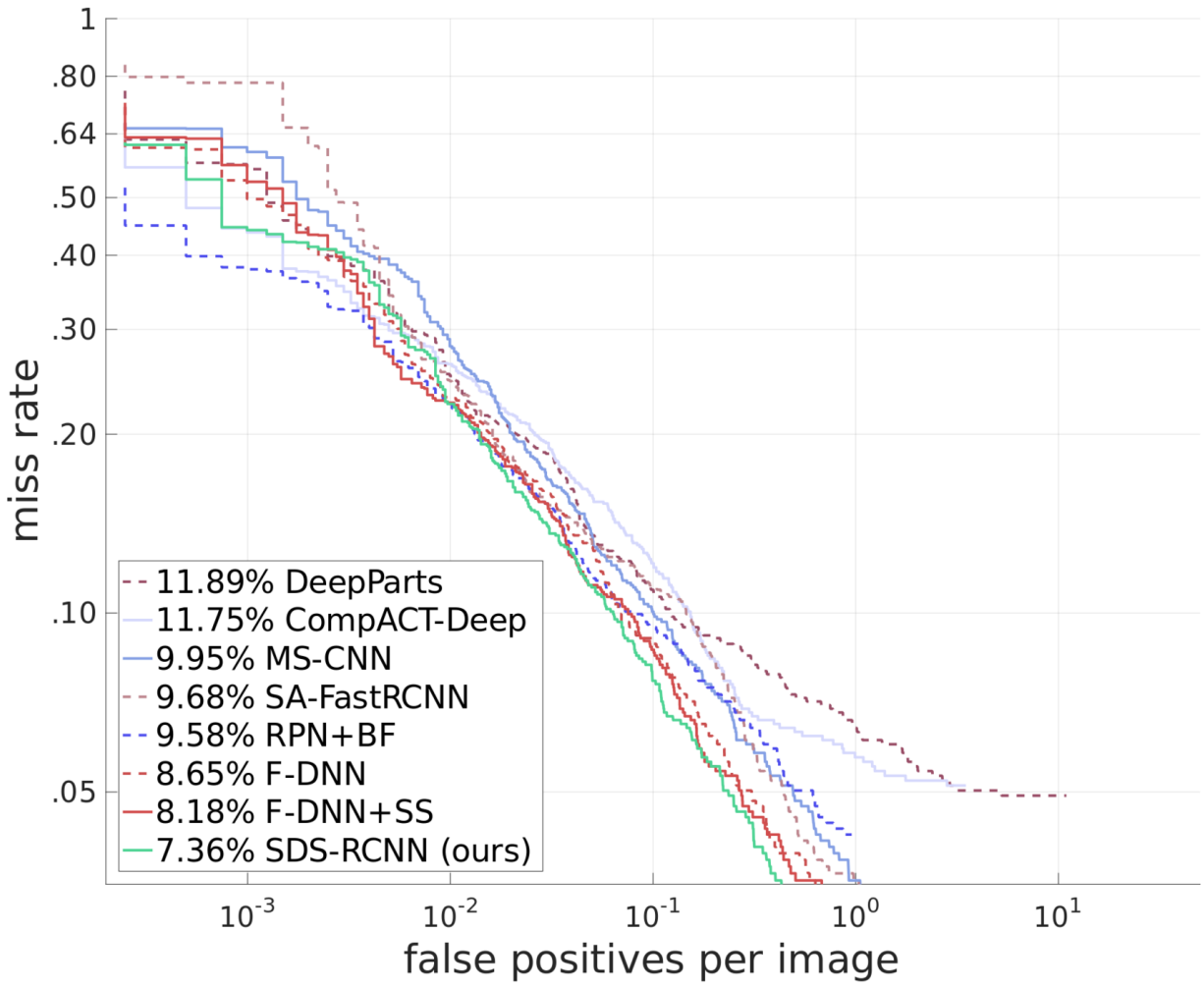

Figure 2.6: Comparison of SDS-RCNN with the state-of-the-art methods on the Caltech dataset using the *reasonable* setting.

2.4.1 Benchmark Comparison

Caltech: The Caltech dataset [38] contains $\sim$350K pedestrian bounding box annotations across 10 hours of urban driving. The log average miss rate sampled against a false positive per image (FPPI) range of $[10^{-2}, 10^{0}]$ is used for measuring performance. A minimum IoU threshold of 0.5 is required for a detected box to match with a ground truth box. For training, we sample from the standard training set according to Caltech10$\times$ [185], which contains 42,782 training images. We evaluate on the standard 4,024 images in the Caltech 1$\times$ test set using the *reasonable* [39] setting,

which only considers pedestrians with at least 50 pixels in height and with less than 35% occlusion.

SDS-RCNN achieves an impressive 7.36% miss rate. The performance gain is a relative improvement of 23% compared to the best published method RPN+BF (9.58%). In Fig. 2.6, we show the ROC plot of miss rate against FPPI for the current top performing methods reported on Caltech.

We further report our performance using just SDS-RPN (without cost-sensitive weighting, Sec. 2.4.3) on Caltech as shown in Table 2.1. The RPN performs quite well by itself, reaching 9.63% miss rate while processing images at roughly $3\times$ the speed of competitive methods. Our RPN is already on par with other top detectors, which themselves contain a RPN. Moreover, the network significantly outperforms other standalone RPNs such as in [183] (14.9%). Hence, the RPN can be leveraged by other researchers to build better detectors in the future.

KITTI: The KITTI dataset [51] contains $\sim$80K annotations of cars, pedestrians, and cyclists. Since our focus is on pedestrian detection, we continue to use only the pedestrian class for training and evaluation. The mean Average Precision (mAP) [43] sampled across a recall range of $[0, 1]$ is used to measure performance. We use the standard training set of $7{,}481$ images and evaluate on the designated test set of $7{,}518$ images. Our method reaches a score of 63.05 mAP on the moderate setting for the pedestrian class. Surprisingly, we observe that many models which perform well on Caltech do not generalize well to KITTI, as detailed in Table 2.1. We expect this is due to both sensitivity to hyperparameters and the smaller training set of KITTI ($\sim 6\times$ smaller than Caltech$10\times$). MS-CNN [13] is the current top performing method for pedestrian detection on KITTI. Aside from the novelty as a multi-scale object detector, MS-CNN augments the KITTI dataset by random cropping and scaling. Thus, incorporating data augmentation could alleviate the smaller training set and lead to better generalization across datasets. Furthermore, as described in the ablation study of Sec. 2.4.3, our weak segmentation supervision primarily improves the detection of unusual shapes and poses (e.g., cyclists, people sitting, bent over). However, in the KITTI evaluation, the person

Method	Caltech	KITTI	Runtime
DeepParts [152]	11.89	58.67	1s
CompACT-Deep [14]	11.75	58.74	1s
MS-CNN [13]	9.95	73.70	0.4s
SA-FastRCNN [91]	9.68	65.01	0.59s
RPN+BF [183]	9.58	61.29	0.60s
F-DNN [41]	8.65	-	0.30s
F-DNN+SS [41]	8.18	-	2.48s
SDS-RPN (ours)	9.63	-	0.13s
SDS-RCNN (ours)	7.36	63.05	0.21s

Table 2.1: Comprehensive comparison of SDS-RCNN with other state-of-the-art methods showing the Caltech miss rate, KITTI mAP score, and runtime performance.

sitting class is ignored and cyclists are counted as false positives, hence such advantages are less helpful.

2.4.2 Efficiency

The runtime performance of SDS-RCNN takes $\sim$0.21s/image. We use images of size 720×960 pixels and a single Titan X GPU for computation. The efficiency of SDS-RCNN surpasses the current state-of-the-art methods for pedestrian detection, often by a factor of $2\times$. Compared to F-DNN+SS [41], which also utilizes segmentation cues, our method executes $\sim 10\times$ faster. The next fastest runtime is F-DNN, which takes 0.30s/image with the caveat of requiring multiple GPUs to process networks in parallel. Further, our SDS-RPN method achieves very competitive accuracy while only taking 0.13s/image ($\sim 3\times$ faster than competitive methods using a single GPU).

Moreover, we further study the trade-off of accuracy and efficiency by designing multiple models meant to be run in real-time while focusing on pedestrians in the "near" category of Caltech [38] (height of ≥ 80px). We first note that our *full* GPU model achieves $< 1\%$ MR on this evaluation setting but runs only at 5 frames per second even while using a high-powered GPU. We begin the

efficiency study by replacing our backbone network with SqueezeNet [69], removing anchors with a scale below the 80px threshold and finally reducing the input image scale to 211px. We observe the performance of the aforementioned model is 4.0% MR on the “near" scale of pedestrians. The model runs at an impressive 48 frames per second on a high-powered CPU and 165 frames per second on a GPU respectively. However, we find that a low-power embedded CPU runs such a model configuration at ≈ 1 frames per second. Therefore, we attempt to further push the runtime efficiency by removing alternating fire modules from SqueezeNet [69], changing the first convolution to have a stride of 16, removing all pooling layers, and reducing the width of all layers with kernels $> 1 \times 1$ by half, which we refer to as SqueezeNet-Embed. After modifying, we pretrain the SqueezeNet-Embed on ImageNet [79] which itself achieves a respectable Top-1% and Top-5% accuracy of 44.9 and 69.2 compared to the peformance of full SqueezeNet [69] which achieves 57.5 and 80.3 respectively. Collectively, this SqueezeNet-Embed model is able to perform at an incredibly fast 387 frames per second on a high-powered CPU at the cost of 27% MR for the “near" scale of pedestrians. In an embedded system with a low-power CPU the speed is is approximately real-time running at ≈ 10 frames per second. Hence, we have demonstrated that our proposed method can be tailored to various application needs varying from real-time on a high-power GPU, high-power CPU, or even a low-power embedded CPU. However, in extreme cases, such as in the case of SqueezeNet-Embed, a high penalty is faced in accuracy.

2.4.3 Ablation Study

In this section, we evaluate how each significant component of our network contributes to performance using the reasonable set of Caltech [38]. First, we examine the impact of four components: weak segmentation supervision, proposal padding, cost-sensitive weighting, and stricter supervision. For each experiment, we start with SDS-RCNN and disable one component at a time as

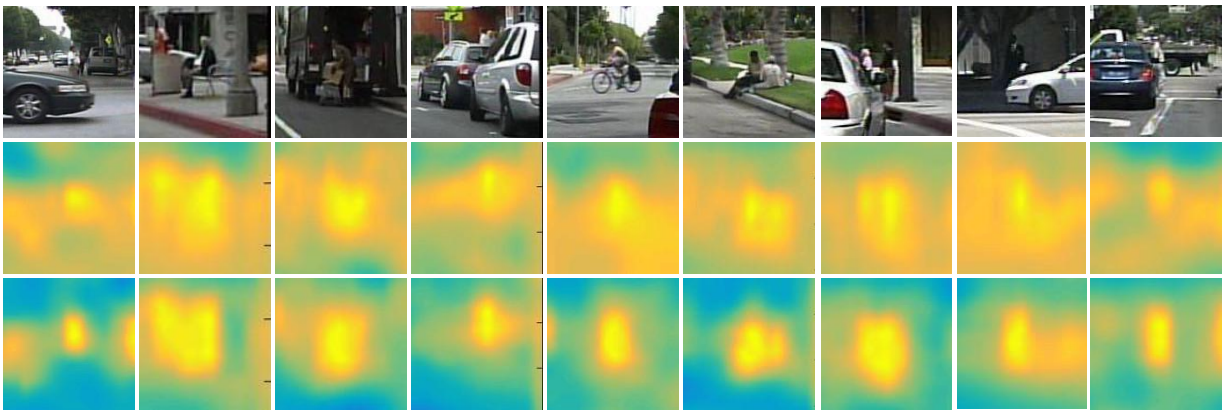

Figure 2.7: Example error sources which are corrected by infusing semantic segmentation into shared layers. Row 1 shows the test images from Caltech1×. Row 2 shows a visualization of the RPN proposal layer using the baseline network which fails on these examples. Row 3 shows a visualization of the proposal layer from SDS-RCNN, which corrects the errors. Collectively, occlusion and *unusual* poses of pedestrians (sitting, cyclist, bent over) make up for 75% of the corrections, suggesting that the the segmentation supervision naturally informs the shared features on robust pedestrian parts and shape information.

Component Disabled	RPN	BCN	Fusion
proposal padding	10.67	13.09	7.69
cost-sensitive	9.63	14.87	7.89
strict supervision	10.67	17.41	8.71
weak segmentation	13.84	18.76	10.41
SDS-RCNN	10.67	10.98	7.36

Table 2.2: Ablation experiments evaluated using the Caltech test set. Each ablation experiment reports the miss rate for the RPN, BCN, and fused score with one component disabled at a time.

summarized in Table 2.2. For simplicity, we disable components globally when applicable. Then we provide detailed discussion on the benefits of stage-wise fusion and comprehensively report the RPN, BCN, and fused performances for all experiments. Finally, since our BCN is designed to not share features with the RPN, we closely examine how sharing weights between stages impacts network diversification and efficiency.

Weak Segmentation: The infusion of semantic features into shared layers is the most critical component of SDS-RCNN. The fused miss rate degrades by a full 3.05% when the segmentation

supervision is disabled, while both individual stages degrade similarly. To better understand the types of improvements gained by weak segmentation, we perform a failure analysis between SDS-RCNN and the “baseline" (non-weak segmentation) network. For analysis, we examine the 43 pedestrian cases which are missed when weak segmentation is disabled, but corrected otherwise. Example error corrections are shown in Fig. 2.7. We find that $\sim 48\%$ of corrected pedestrians are at least partially occluded. Further, we find that $\sim 28\%$ are pedestrians in *unusual* poses (e.g., sitting, cycling, or bent over). Hence, the feature maps infused with semantic features become more robust to atypical pedestrian shapes. These benefits are likely gained by semantic segmentation having indiscriminant coverage of all pedestrians, unlike object detection which requires specific alignment between pedestrians and anchor shapes. A similar advantage could be gained for object detection by expanding the coverage of anchors, but at the cost of computational complexity.

Proposal Padding: While padding proposals is an intuitive design choice to provide background context (Fig. 2.3), the benefit in practice is minor. Specifically, when proposal padding is disabled, the fused performance only worsens from 7.36% to 7.69% miss rate. Interestingly, proposal padding remains critical for the individual BCN performance, which degrades heavily from 10.98% to 13.09% without padding. The low sensitivty of the fused score to padding suggests that the RPN is already capable of localizing and differentiating between partial and full-pedestrians, thus improving the BCN in this respect is less significant.

Cost-sensitive: The cost-sensitive weighting scheme used to regularize the importance of large pedestrians over small pedestrians has an interesting effect on SDS-RCNN. When the cost-sensitive weighting is disabled, the RPN performance actually improves to an impressive 9.63% miss rate. In contrast, without cost-sensitive weighting the BCN degrades heavily, while the fused score degrades mildly. A logical explanation is that imposing a precedence on a single scale is counter-

intuitive to the RPN achieving high recall across *all* scales. Further, the RPN has the freedom to learn scale-dependent features, unlike the BCN which warps to a fixed size for every proposal. Hence, the BCN can gain significant boost when encouraged to focus on large pedestrian features, which may be more scale-independent than features of small pedestrians.

Strict Supervision: Using a stricter labeling policy while training the BCN has a substantial impact on the performance of both the BCN and fused scores. Recall that the strict labeling policy requires a box to have IoU > 0.7 to be considered foreground, while the standard policy requires IoU > 0.5. When the stricter labeling policy is reduced to the standard policy, the fused performance degrades by 1.35%. Further, the individual BCN degrades by 6.43%, which is on par with the degradation observed when weak segmentation is disabled. We examine the failure cases of the strict versus non-strict BCN and observe that the false positives caused by double detections reduce by $\sim 22\%$. Hence, the stricter policy enables more aggressive suppression of poorly localized boxes and therefore reduces double detections produced as localization errors of the RPN.

Stage Fusion: The power of stage-wise fusion relies on the assumption that the each network will diversify in their classification characteristics. Our design explicitly encourages this diversification by using separate labeling policies and training distributions for the RPN and BCN. Table 2.2 shows that although fusion is useful in every case, it is difficult to anticipate how well any two stages will perform when fused without examining their specific strengths and weaknesses.

To better understand this effect, we visualize how fusion behaves when the RPN and BCN disagree (Fig. 2.8). We consider only boxes for which the RPN and BCN disagree using a decision threshold of 0.5. We notice that both networks agree on the majority of boxes (~ 80K), but observe an interesting trend when they disagree. The visualization clearly shows that the RPN tends to predict a significant amount of background proposals with high scores, which are corrected after

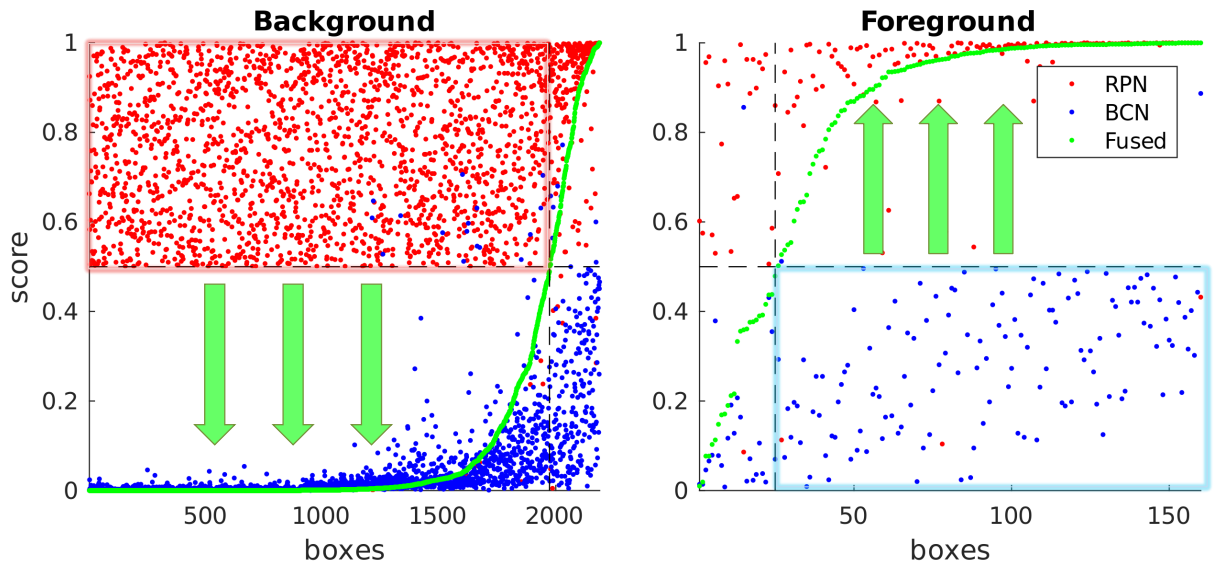

Figure 2.8: Visualization of the diversification between the RPN and BCN classification characteristics. We plot only boxes which the RPN and BCN of SDS-RCNN disagree on using a threshold of 0.5. The BCN drastically reduces false positives of the RPN, while the RPN corrects many missed detections by the BCN.

being fused with the BCN scores. The inverse is true for disagreements among the foreground, where fusion is able to correct the majority of pedestrians boxes given low scores by the BCN. It is clear that whenever the two networks disagree, the fused result tends toward the true score for more than $\sim 80\%$ of the conflicts.

Sharing Features: Since we choose to train a separate RPN and BCN, without sharing features,

Shared Layer	BCN MR	Fused MR	Runtime
conv5	16.24	10.87	0.15s
conv4	15.53	10.42	0.16s
conv3	14.28	8.66	0.18s
conv2	13.71	8.33	0.21s
conv1	14.02	8.28	0.25s
RGB	10.98	7.36	0.21s

Table 2.3: Stage-wise sharing experiments which demonstrate the trade-off of runtime efficiency and accuracy, using the Caltech dataset. As sharing is increased from RGB (no sharing) to conv5, both the BCN and Fused miss rate (MR) become less effective.

we conduct comprehensive experiments using different levels of stage-wise sharing in order to understand the value of diversification as a trade-off to efficiency. We adopt the Faster R-CNN feature sharing scheme with five variations differing at the point of sharing (conv1-5) as detailed in Table 2.3. In each experiment, we keep all layers of the BCN except those before and including the shared layer. Doing so keeps the effective depth of the BCN unchanged. For example, if the shared layer is conv4 then we replace conv1-4 of the BCN with a RoIPooling layer connected to conv4 of the RPN. We configure the RoIPooling layer to pool to the resolution of the BCN at the shared layer (e.g., conv4 $\rightarrow 14 \times 14$, conv5$\rightarrow 7 \times 7$).

We observe that as the amount of sharing is increased, the overall fused performance degrades quickly. Overall, the results suggest that forcing the networks to share feature maps lowers their freedom to diversify and complement in fusion. In other words, the more the networks share the more susceptible they become to redundancies. Further, sharing features up to conv1 becomes slower than no stage-wise sharing (e.g., RGB). This is caused by the increased number of channels and higher resolution feature map of conv1 (e.g., $720 \times 960 \times 64$), which need to be cropped and warped. Compared to sharing feature maps with conv3, using no sharing results in a very minor slow down of 0.03 seconds while providing a 1.30% improvement to miss rate. Hence, our network design favors maximum precision for a reasonable trade-off in efficiency, and obtains speeds generally $2\times$ faster than competitive methods.

2.5 Summary

We present a multi-task infusion framework for joint supervision on pedestrian detection and semantic segmentation. The segmentation infusion layer results in more sophisticated shared feature maps which tend to *illuminate* pedestrians and make downstream pedestrian detection easier. We

analyze how infusing segmentation masks into feature maps helps correct pedestrian detection errors. In doing so, we observe that the network becomes more robust to pedestrian poses and occlusion compared to without. We further demonstrate the effectiveness of fusing stage-wise scores and encouraging network diversification between stages, such that the second stage classifier can learn a stricter filter to suppress background proposals and become more robust to poorly localized boxes. In our SDS-RCNN framework, we report new state-of-the-art performance on the Caltech pedestrian dataset (23% relative reduction in error), achieve competitive results on the KITTI dataset, and obtain an impressive runtime approximately $2\times$ faster than competitive methods.

Chapter 3

Pedestrian Detection with Autoregressive Network Phases

We present an autoregressive pedestrian detection framework with cascaded phases designed to progressively improve precision. The proposed framework utilizes a novel lightweight stackable decoder-encoder module which uses convolutional re-sampling layers to improve features while maintaining efficient memory and runtime cost. Unlike previous cascaded detection systems, our proposed framework is designed within a region proposal network and thus retains greater context of nearby detections compared to independently processed RoI systems. We explicitly encourage increasing levels of precision by assigning strict labeling policies to each consecutive phase such that early phases develop features primarily focused on achieving high recall and later on accurate precision. In consequence, the final feature maps form more peaky radial gradients emulating from the centroids of unique pedestrians. Using our proposed autoregressive framework leads to new state-of-the-art performance on the reasonable and occlusion settings of the Caltech pedestrian dataset, and achieves competitive state-of-the-art performance on the KITTI dataset.

3.1 Introduction

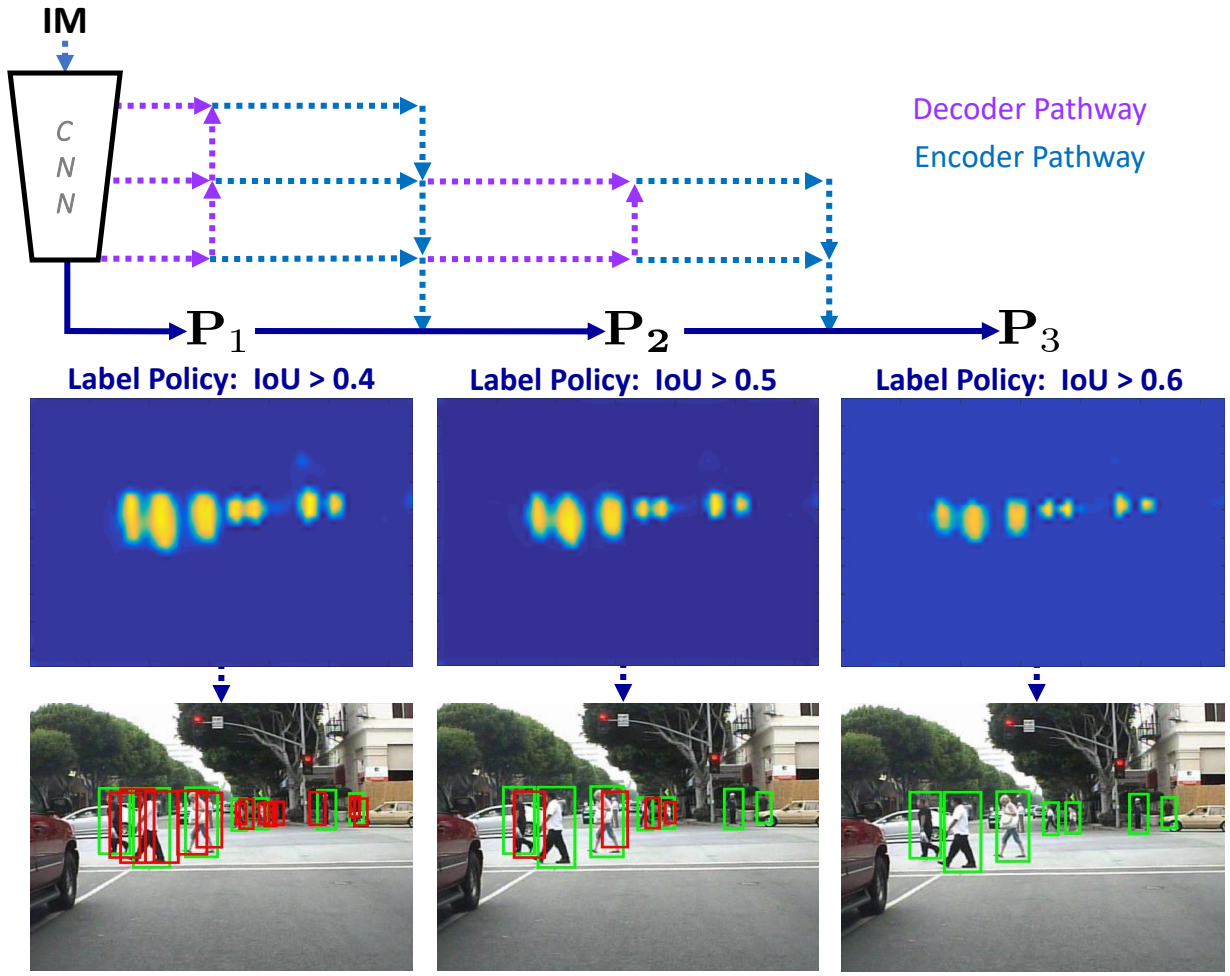

Figure 3.1: Illustration of our proposed autoregressive framework with sample phase ($P_{1\rightarrow 3}$) classification prediction maps and box visualizations under Caltech [38] dataset. Our method iteratively re-scores predictions under incrementally more precise label policies, using a series of de-encoder modules comprised of decoder and encoder pathways. Notice a heavy reduction in false positives (red) as phases progress, while true positives (green) are retained.

Detecting pedestrians in urban scenes remains to be a challenge in computer vision despite recent rapid advances [11, 75, 108, 131, 147, 160, 186, 187, 189]. The use of ensemble [41, 150, 180] and recurrent [131, 148] networks has been successful in top-performing approaches of pedestrian / object detection. Recurrent networks refine upon their own features while ensemble networks

gather features through separate deep classifiers. Both techniques offer a way to obtain stronger and more robust features, thus better detection.

However, the characteristics of ensemble and recurrent networks are distinct. Ensemble networks assume that separate networks will learn diversified features which when combined will become more robust. In contrast, recurrent networks inherit previous features as input while further sharing weights between successive networks. Hence, recurrent networks are more capable of refining than diversifying. Intuitively, we expect that both feature *diversification* and *refinement* are important components to pair together.

Therefore, we explore how to approximate an ensemble of networks using a stackable decoder-encoder module and incorporating an *autoregressive*[1] flow to connect them, as illustrated in Fig. 3.1. We formulate our framework as a series of phases where each is a function of the previous phase feature maps and classification predictions. Our decoder-encoder module is made of bottom-up and top-down pathways similar to [78, 98, 106, 119]. However, rather than using bilinear or nearest neighbor re-sampling followed by conventional convolution, we propose memory-efficient convolutional re-sampling layers to generate features and re-sample *simultaneously* in a single step.

In essence, our approach aims to take the best world of both the ensemble and recurrent approaches. For instance, since past predictions and features are re-used, our network is able to refine features when necessary. Secondly, since our phases incorporate inner-lateral convolutions and do *not* share weights, they are also capable to learn new and diversified features. Furthermore, we are able to design the network with an efficient overhead due to the added flexibility of using non-shared network weights for each phase and by using memory-efficient convolutional re-sampling layers. As a consequence, we are able to choose optimal channel settings with respect to efficiency

[1] We adopt naming distinction of *autoregressive* (vs. recurrent) as a network conditioned on previous predictions without the constraint of repeated shared weights, inspired by terminology in WaveNet [157] which uses casual convolution instead of conventional recurrence.

and accuracy.

To take full advantage of the autoregressive nature of our network, we further assign each phase a distinct labeling policy which iteratively becomes more *strict* as phases progress. In this way, we expect that the predictions of each consecutive phase will become less noisy and produce tighter and more clusterable prediction maps. Under the observation that our proposed autoregressive region proposal network (RPN) obtains a high recall in the final phase, we also incorporate a simple hard suppression policy into training and testing of our second-stage R-CNN classifier. Such a policy dramatically narrows the subset of proposals processed in the second-stage pipeline ($\sim 65\%$), and greatly alleviates the runtime efficiency accordingly.

We evaluate our framework on the Caltech [38] pedestrian detection dataset under challenging occlusion settings, using both the original and newly proposed [184] annotations, and further on the KITTI [51] benchmark. We achieve state-of-the-art performance under each test setting and report a marginal overhead cost in runtime efficiency.

To summarize, our contributions are the following:

- We propose a multi-phase autoregressive pedestrian detection system *inside* a RPN, where each phase is trained using increasingly precise labeling policies.
- We propose a lightweight decoder-encoder module to facilitate feature map refinement and message passing using convolutional re-sampling layers for memory-efficient pathways.
- We achieve state-of-the-art performance on Caltech [38] under various challenging settings, and competitive performance on KITTI [51] pedestrian benchmark.

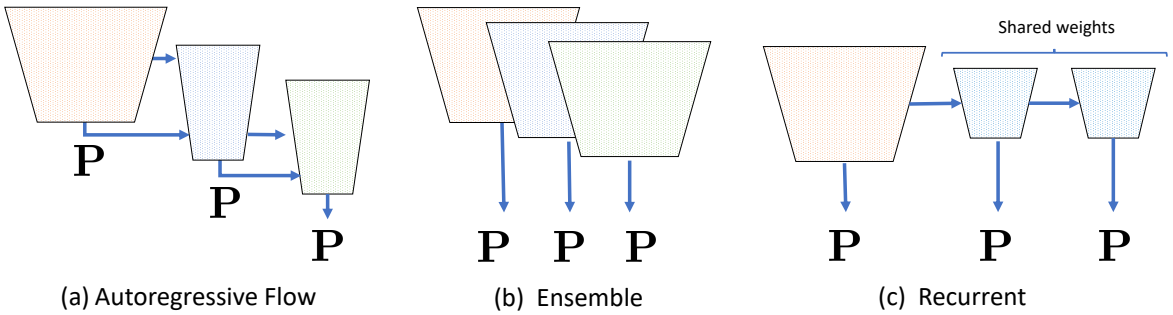

Figure 3.2: Predictions of our autoregressive network (a) are directly conditioned on past feature maps as recurrent network (c) and do not share weights between phases as ensemble network (b). Unlike either, our network is further conditioned on past predictions.

3.2 Related Work

Ensemble Networks: Recent top-performing methods [41, 150, 180] on detection have employed ensemble-based techniques where predictions from multiple deep convolutional neural networks (CNNs) are fused. For instance, [41] propose a soft-weighting scheme using an ensemble of independent detectors, which demonstrate high accuracy with fused scores. However, one drawback is having multiple CNNs in memory and processing each in parallel. Thus, both the scalability as networks become larger and usability in memory-constrained systems are lessened. Further, [11] form a small ensemble by fusing RPN scores with the scores of a R-CNN detector and demonstrates improved performance. Compared to these methods, our *single* RPN functions as an ensemble of inter-connected small networks, which can improve the precision without critically obstructing runtime or memory efficiency.

Cascaded Networks: A similar line to ensemble networks take form of cascaded detection systems [15, 123, 128], which build on a series of R-CNN detectors and function on cropped region-of-interests (RoIs) generated by a static proposal network. In contrast, our work focuses as a *fully convolutional* cascade inside a proposal network. Therefore, our network is more equipped to

utilize contextual cues of surrounding detections to inform suppression of duplicate detections, whereas cropped RoIs are processed *independently* of other proposals. Liu *et al.* [108] propose supervision using incremental labeling policies similar to our approach. However, rather than making immediate predictions based only on previous predictions, we develop *new* features through our decoder-encoder pathway.

Recurrent Networks: Recurrent networks are a powerful technique in many challenging procedural [59, 115] and temporal [17, 137, 146] computer vision problems. Recently, it has been further demonstrated in urban object detection [131] and person head detection [148]. For instance, [148] uses recurrent LSTM to iteratively detect a single person at a time until reaching an end condition, thus side-stepping the need to perform non-maximum suppression (NMS) in post. In contrast, [131] proposes a rolling recurrent convolution (RRC) model which refines feature maps and produces new detections at each step. From this respect, our proposed method is similar to RRC, but with two critical differences. Firstly, the networks of our phases are not shared. This enables us to learn specialized (ensemble-like) features in each phase and gives more freedom in network design of a phase, which may aid runtime efficiency when using conservative designs. Secondly, we base each phase conditioned on previous feature maps *and* predictions, which form a more potent autoregressive foundation. We show a high-level comparison of our autoregressive network, ensemble networks, and recurrent networks in Fig. 3.2.

Encoder-Decoder Networks: Many recent works [78, 98, 106, 134] have explored multi-strided feature maps re-use within computer vision. Each variant of architectures utilize a series of convolution, feature aggregation (concat, residual), and up-sampling / pooling layers in order to form an encoder-decoder structure. Similar to the network structure in [119] for human pose estimation, we incorporate stackable top-down *and* bottom-up pathways. However, in contrast to prior work, we

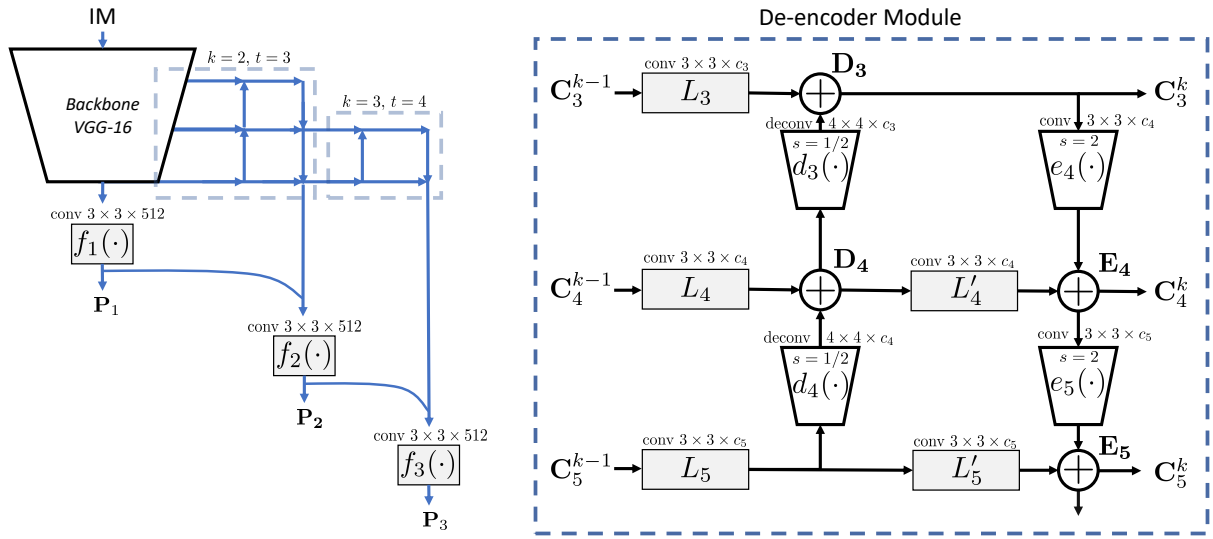

Figure 3.3: Overview of our proposed AR-RPN framework (left) and detailed illustration of our de-encoder module (right). The de-encoder module consist of top-down and bottom-up pathways with inner-lateral convolution between pathways to produce diversified features, as well as convolutional re-sampling layers (s denotes convolutional stride) e_i and d_i for memory-efficient feature generation. We further condition predictions on the previous phase predictions through concatenation within $f_k(\cdot)$.

design our de-encoder module without explicitly using bilinear or nearest neighbor re-sampling. Instead, we uniquely blend the feature generation and re-sampling into a single convolution layer using a fractional stride ($\uparrow$) or strided convolution ($\downarrow$), making the travel nodes in our streams as *compact* as possible. We show in ablation that a single convolutional re-sampling layer consumes low memory and performs better compared to the conventional two-step techniques previously used.

3.3 Autoregressive Detector

Our proposed framework is made up of two stages: an autoregressive RPN hence referred to as AR-RPN, and a second-stage R-CNN detector each founded on [132, 183]. We collectively refer to both stages as AR-Ped. As shown in Fig. 3.3, AR-RPN consists of multiple phases, where each

predicts classification scores and passes these predictions *and* their features into the next phase. Each phase is connected to the last through a bottom-up and top-down pathways, which form a lightweight decoder-encoder module. This module is stackable onto the backbone RPN and onto itself repeating. We supervise each phase to jointly learn increasingly more precise predictions by imposing a stricter labeling policy to consecutive phases, thereby producing more peaky and clusterable classifications in the final phase. We apply the box transformations, NMS, and a hard suppression policy to the final predictions for which the remaining subset are used to train a specialized R-CNN detector.

3.3.1 De-Encoder Module

To perform autoregressive detection in a *single* model, we design a stackable decoder-encoder module, termed *de-encoder*, where its top-down pathway leverages past feature maps and its bottom-up pathway encodes stronger semantics. Following [98], we give each pathway the ability to learn from feature maps at multiple depths of the backbone network. Importantly, our design encourages the highest level features to remain at the lowest resolution where object detection functions most efficiently. Intuitively, the de-encoder enables the network to look back at previous features *and* learn more advanced features during re-encoding.

Let us recall that typical network architectures, e.g., VGG-16 [141] and ResNet-50 [65], function from low to high stride levels using a series of convolution and pooling layers. We denote the set of strides of a backbone network as $\mathbf{S}$, where 2^{i-1} is the down-sampling factor of the ith stride level preceding a pooling operation. In pedestrian detection, it is common to have $n = 5$ unique stride levels such that $\mathbf{S} = \{1, 2, 4, 8, 16\}$. The hyperparameters of the de-encoder module include a designated target stride t and channel width c_i specific to each stride, which respectively control how far up in resolution the phase should de-encode and how many channels at each stride should

be learned.

The primary goal of the de-encoder module is to produce finer features at each level from the target stride t to the final stride n of the network. Denoting $\mathbf{C}^k = \{\mathbf{C}^k_t, \ldots, \mathbf{C}^k_n\}$ as the refined features of the kth phase at each stride, $g_k(\cdot)$ as the set of convolutional and ReLU layers, Φ_k the respective weights, and t_k the target stride of feature maps to de-encode and refine, the autoregressive nature of the feature generation can be expressed as:

$$\mathbf{C}^k = g_k(\mathbf{C}^{k-1} \mid \Phi_k,\ t_k). \tag{3.1}$$

Hence, each phase of the network takes as input the previous phase feature maps and produces more advanced features. Initial features $\mathbf{C}^1$ are given from top-most layers at corresponding strides from the backbone (e.g., in VGG-16 $\mathbf{C}^1_4$ = conv4_3, $\mathbf{C}^1_5$ = conv5_3, and so forth).

Top-down pathway: We design our top-down decoder for phase k by attaching a convolutional layer with BN [70] to $\{\mathbf{C}^{k-1}_{t\ldots n}\}$ feature maps, which produce inner-lateral convolutions $\mathbf{L}_i$ with corresponding channel widths c_i. Rather than using a two-step process comprised of a bilinear / nearest neighbor up-sampling followed by convolution as done in prior work, we denote $d_i(\cdot)$ as a convolutional up-sampling layer which *simultaneously* performs $2\times$ up-sampling and feature reduction into channel width c_i using fractionally strided convolution. The combined operation is more efficient in both memory and runtime. Starting with the highest feature stride n, we use $d_i(\cdot)$ to iteratively decode features, which are then fused with the lateral features at the decoded stride $\mathbf{L}_i$ through element-wise addition, denoted:

$$\mathbf{D}_i = d_i(\mathbf{D}_{i+1}) + \mathbf{L}_i. \tag{3.2}$$

We begin with the base case of $\mathbf{D}_n = \mathbf{L}_n$, and repeat this procedure until the target stride feature map $\mathbf{D}_t$ is reached. In theory, the top-down pathway enables high-level semantics to be passed down through the decoded term $d_i(\mathbf{D}_{i+1})$ and low-level features to be re-examined using $\mathbf{L}_i$.

Bottom-up pathway: We design the bottom-up encoder in the opposite manner as the decoder. We first attach a convolutional layer with BN to each $\{\mathbf{D}_{t+1\ldots n}^{k-1}\}$ which each produce *new* laternal features $\mathbf{L}_i'$ with c_i channels. Similar to the decoder pathway, we denote $e_i(\cdot)$ as a single convolutional down-sampling layer which *simultaneously* performs $2\times$ down-sampling and feature expansion into channel width c_i using strided convolution, rather than conventional two-step process used in previous work. We use $e_i(\cdot)$ to iteratively encode the features at each stride, which are then fused with the lateral features of the encoded stride $\mathbf{L}_i'$ via element-wise addition, denoted as:

$$\mathbf{E}_i = e_i(\mathbf{E}_{i-1}) + \mathbf{L}_i'. \tag{3.3}$$

As the name suggests, the bottom-up encoder starts with the lowest stride t and repeats until the nth stride is reached, such that lateral features at t is $\mathbf{E}_t = \mathbf{D}_t$. The bottom-up pathway enables the network to encode low-level features from the lowest stride through the $e_i(\mathbf{E}_{i-1})$ term and for higher-level features to be re-examined using $\mathbf{L}_i'$.

3.3.2 Autoregressive RPN

We utilize the standard RPN head and multi-task loss proposed in [54] following the practices in [183]. We predefine a set of anchor shapes which act as hyperparameters describing the target pedestrian scales. The RPN head is comprised of a proposal feature extraction (PFE) layer connected to two sibling layers which respectively predict anchor classification (cls) and bounding box regression (bbox) output maps, hence forming a multi-task learning problem.

Multi-phase Network: Our RPN is comprised of a total of $N_k = 3$ phases. The first phase is simply the backbone network starting with the modified VGG-16 [141] that has strides of $\mathbf{S} = \{1, 2, 4, 8, 16\}$. The second phase is a de-encoder module which has a target stride $t = 3$ and channel widths of $c_3 = 128, c_4 = 256, c_5 = 512$. The final phase is another stack of the de-encoder module following the same channel settings but uses a memory conservative lower target stride of $t = 4$. The spatial resolution at ith stride can be denoted as $w_i \times h_i = \frac{W}{2^{i-1}} \times \frac{H}{2^{i-1}}$, where $W \times H$ is the input image resolution. Thus, the final proposal network architecture forms a stair-like shape as in Fig. 3.3.

Autoregressive Flow: To enable the autoregressive flow between phases, we place a PFE layer and classification layer at the end of each phase encoder. For all phases except the first, we *concatenate* the previous phase predictions into the input features for the corresponding phase PFE layer. In doing so, each phase is able to start with strong compact features by directly utilizing its previous phase predictions. Further, the PFE layer of the final phase N_k produces the bounding box regression output map, since these features are the most precise and *peaky* within the network.

Formally, we denote functions $f_k(\cdot)$ and $p_k(\cdot)$ as the kth phase PFE layers and classification layers respectively. We build $f(\cdot)$ as a convolutional layer with 3×3 kernel and 512 output channels followed by a ReLU layer, while $p(\cdot)$ a convolutional layer with 1×1 kernel and outputs channels $2\times$ the number of anchors (A). Thus, $p_k(\cdot)$ forms an autoregressive function of previous phase predictions with an output dimension of $w_5 \times h_5 \times 2A$, via:

$$\mathbf{P}_k = p_k(f_k(\mathbf{P}_{k-1} \parallel \mathbf{C}_n^k)), \tag{3.4}$$

where $\mathbf{P}_{k-1}$ is the classification feature map of the previous phase, aka, past predictions, $\parallel$ is the concatenation operator, and $\mathbf{C}_n^k$ is the last encoded feature map of the kth phase. As defined, the

PFE $f_k(\cdot)$ and classification layer $p_k(\cdot)$ are conditioned autoregressively on past predictions which logically act as compact but powerful semantic features. In this way, each phase is more free to learn new features $\mathbf{C}_n^k$ to directly *complement* the past predictions. In essence, the autoregressive flow can be seen as running memory of the most compact and strong features within the network.

Classification Task: Each classification layer which proceeds a PFE layer is formulated as proposed in [132] following experimental settings of [11]. Formally, given a PFE layer with dimensions $w \times h$, the designated classification layer predicts a score for *every* spatial location of the image $(x, y) \in \mathbb{R}^{w \times h}$ against every predefined anchor shape $a \in \mathbf{A}$, and every target class. Every spatial location of the prediction map is therefore treated as a distinct box with its own corresponding classification score. To produce labels for each box, a labeling policy is adopted using a hyperparameter h that controls the box criteria of Intersection over Union (IoU) with ground truths in order to be considered foreground. After every box is assigned a label according to the labeling policy, each classification layer is supervised using multinomial cross-entropy logistic loss as in [54].

Localization Task: The localization task is formed using the same set of anchor boxes described in the classification task. The localization task aims to perform bounding box regression that predicts a bounding box transformation for each foreground box towards the nearest pedestrian. A proposal box is considered *nearby* a pedestrian ground truth if there is at least h intersection over union between the two boxes. The box transformation is defined by 4 variables consisting of translation (t_x, t_y) and scale factors (t_w, t_h) such that when applied will transform the source box into the target ground truth. We train the bounding box regression values using Smooth L_1 loss [54].

Incremental Supervision: In order to better leverage the autoregressive and de-encoder proper-

ties of AR-RPN, we choose to assign *different* classification labeling policies onto each consecutive phase. We emphasize that the de-encoder modules enable the network to adapt and become a stronger classifier, which can be exploited to produce more accurate and tighter classification clusters when supervised with incrementally stricter labeling policies.

Let us briefly discuss the trade-offs regarding different labeling policies. Consider using a labeling policy of $h = 1$, which is approximately equivalent to requiring the network output a *single* box for each pedestrian and thus the imbalance of classes may be difficult. In contrast, as a labeling policy becomes more lenient at $h = 0.5$, the classification becomes more balanced but produces many false positives as duplicate detections. In theory, bounding box regression will reduce the impact of double detections by transforming boxes into clusters which can be suppressed by NMS. Ideally, a network has either high-performing bounding box regression and/or tight clusterable classification maps, since both enable NMS to cluster duplicate detections. Therefore, rather than using a single discrete labeling policy of $h = 0.5$, we assign lenient-strict policies $h_1 = 0.4,\ h_2 = 0.5,\ h_3 = 0.6$, to each phase classification layer respectively. In contrast to [108], we enforce incremental supervision between de-encoder modules rather than being applied immediately in quick succession. In consequence, our classification score maps are supervised to gradually become more peaky and clusterable.

Loss Formulation: In addition to the classification and bounding box regression losses, we further add auxiliary losses in the form of weak semantic segmentation as in [11]. Specifically, during training we add a binary semantic segmentation layer to each stride of the first top-down pathway to act as an auxiliary loss and accelerate training. We formally define the joint loss terms incorporating phase classification softmax loss L_{cls}, final phase localization Smooth L_1 loss L_{bbox},

	Caltech Reasonable				Caltech Occlusion			KITTI		
	MR^{O}_{-2}	MR^{O}_{-4}	MR^{N}_{-2}	MR^{N}_{-4}	Partial^{O}	Heavy^{O}	RT (ms)	Easy	Mod.	Hard
MS-CNN [13]	9.95	22.45	8.08	17.42	19.24	59.94	64	*83.92*	*73.70*	**68.31**
RRC [131]	–	–	–	–	–	–	75	–	**75.33**	–
RPN+BF [183]	9.58	18.60	7.28	16.76	24.23	74.36	88	75.58	61.29	56.08
F-DNN [41]	8.65	19.92	6.89	*14.75*	15.41	*55.13*	–	–	–	–
TLL(MRF)+LSTM [147]	7.40	–	–	–	–	–	–	–	–	–
ALFNet [108]	–	–	6.10	–	–	–	–	–	–	–
SDS-RCNN [11]	*7.36*	*17.82*	6.44	15.76	14.86	58.55	95	–	63.05	–
RepulsionLoss [160]	–	–	*5.00*	–	–	–	–	–	–	–
FRCNN+ATT-vbb [187]	10.33	–	–	–	–	45.18	–	–	–	–
PDOE+RPN [189]	7.60	–	–	–	*13.30*	44.40	–	–	–	–
GDFL [96]	7.85	19.86	–	–	16.74	*43.18*	–	**84.61**	68.62	66.86
DSSD [48]+Grid [75]	10.85	18.20	–	–	24.28	**42.42**	–	–	–	–
AR-RPN (ours)	8.01	21.62	5.78	15.86	16.30	58.06	86	–	–	–
AR-Ped (ours)	**6.45**	**15.54**	**4.36**	**11.39**	**11.93**	48.80	91	83.66	73.44	*68.12*

Table 3.1: Comprehensive comparison of our frameworks and the state-of-the-art on the Caltech and KITTI benchmarks, in both accuracy and runtime (RT). We show the Caltech miss rates at multiple challenging settings, with both the original (O) and new (N) annotations, and at occlusion settings with the original annotations and FPPI range MR^{O}_{-2}. Further, we evaluate the KITTI pedestrian class under easy, moderate, and hard settings, with mean Average Precision (mAP) [51]. **Boldface**/*italic* indicate the best/second best performance.

and each softmax auxiliary loss L_{seg} as:

$$L = \sum_{k=1}^{N_k} \lambda_k L_{cls} + \lambda_b L_{bbox} + \lambda_s \sum_{i=3}^{5} L_{seg}, \tag{3.5}$$

where k corresponds to phases $1 \rightarrow N_k$ of the full network, and i represents stride for each auxiliary segmentation layer of the backbone network. We use Caffe [71] with SGD following the settings in [183] in our training. We set $\lambda_1 = \lambda_2 = 0.1, \lambda_3 = 1$, $\lambda_b = 5$, and $\lambda_s = 1$.

3.3.3 R-CNN Detector

Most pedestrian detection frameworks are derivatives of Faster R-CNN [132], and hence incorporate a second-stage scale-invariant region classifier termed as R-CNN. Following [11], we utilize a modified VGG-16 as a R-CNN that functions on cropped RGB regions proposed by AR-RPN,

utilizes a strict labeling policy, and fuses its scores with the RPN. However, unlike past methods we impose a simple hard suppression policy that suppresses all box proposals with a score less than a hyperparameter z. This has two advantages. Firstly, it greatly improves runtime since only a subset of proposals need to be processed. Secondly, by *focusing* on only the hard samples leftover from the RPN, the R-CNN learns specialized classification similar to the motivation of the AR-RPN.

Loss Formulation: As in the AR-RPN, we also use softmax loss to train the R-CNN. We use a strict labeling policy requiring $h \geq 0.7$ IoU for foreground, a weak segmentation auxiliary loss L_{seg}, and height sensitive weighting scheme w as detailed in [11]. We set $z = 0.005$ to impose a score suppression of the RPN proposals and eliminate confident background proposals from being re-processed. In practice, the suppression dramatically reduces the search space for both efficiency and accuracy while critically keeping recall *unaffected*. Thus, we denote the R-CNN loss as:

$$L = \sum_{j} w_j L_{cls}(c_j, \hat{c_j}) + L_{seg}, \quad \text{if } c_j \geq z, \tag{3.6}$$

where j corresponds to each proposal of AR-RPN, c is the classification result of the R-CNN, and $\hat{c}$ is the class label. We use Caffe to train the R-CNN following settings of [11].

3.4 Experiments

We evaluate our proposed AR-Ped framework on two challenging datasets: Caltech [38, 39] and KITTI [51]. We perform experiments ablating our approach from the perspective of design choices and hyperparameters. We further examine the qualitative changes and analyze the quantitative peakiness in detections across phases.

3.4.1 Caltech

The Caltech [38, 39] dataset is a widely used benchmark on pedestrian detection that contains 10 hours of video taken from an urban driving environment with $\sim 350{,}000$ bounding box annotations and $2{,}300$ unique pedestrians. We use the Caltech$10\times$ for training and the Caltech reasonable setting [39] for testing, unless otherwise specified. The evaluation uses a miss rate (MR) metric averaged over a false positive per image (FPPI) range of $[10^{-2}, 10^{0}]$ and also a more challenging metric over the range $[10^{-4}, 10^{0}]$, respectfully referred to as MR_{-2} and MR_{-4}. Recently, new annotations are released [184] to correct the official annotations in terms of consistency and box alignment. For completeness, we evaluate on both the original and the new annotations, denoted respectively as MR^{O} and MR^{N}.

We compare our work to the state-of-the-art pedestrian detection methods of Caltech with respect to the core experimental configurations of using each combination of original/FPPI setting, and partial/heavily occlusion within the original annotation space as defined in [38]. We limit our comparison to the top-2 methods of any sub-category trained using Caltech$10\times$ dataset since these comprise the most highly competitive methods. We also emphasize that we are among the few methods to *comprehensively* evaluate and report each setting and insist to open-source our code to the community upon release.

Our method advances the state-of-the-art on all but one evaluation setting, as detailed in Table 3.1. Under the most common benchmark reasonable setting, we achieve a miss rate of 6.45% ($\downarrow 0.91$) and 4.36% ($\downarrow 0.64$) on the official annotations MR^{O}_{-2} and new annotation MR^{N}_{-2} respectively. Further, our approach has increased robustness to partial occlusion ($\downarrow 1.37\%$ miss rate). Compared to methods which do not explicitly address occlusion [11, 13, 41, 183], our method also improves w.r.t heavy occlusion ($\downarrow 6.33\%$ miss rate). Yet, our method underperforms on heavy oc-

clusion compared to work specially designed to target occlusion problem [75, 96, 160, 189], which is orthogonal to our work.

We further produce a runtime analysis for state-of-the-art works with public code using the *same* controlled machine with NVIDIA 1080 Ti GPU, as summarized in Table 3.1. Our method retains a competitive runtime efficiency due to the light overhead design of our de-encoder module while still improving accuracy in all but one setting.

3.4.2 KITTI

KITTI is a popular urban object detection dataset which offers annotations for cars, pedestrians and cyclists. We use the official training set of $7{,}481$ images and evaluate on the standard $7{,}518$ test images. We adopt the settings and core training code of [13] in order to initialize good starting hyperparameters. However, due to GPU memory constraints we set the input image scale to 576 height resolution and achieve competitive performance on the pedestrian class, as reported in Table 3.1. As described in [11], high performing pedestrian detectors [11, 91, 183] on Caltech and KITTI do not usually have high correlation. We emphasize that *our AR-Ped is among the first to report high performance for both datasets*, which suggests the generalization of our model to pedestrian detection rather than a specific dataset.

3.4.3 Ablations

All ablation experiments use our AR-RPN and the Caltech test set under the reasonable MR^{O}_{-2} FPPI setting, as this is the most widely tested setting on Caltech.

What are optimal de-encoder settings? In order to analyze the de-encoder module, we ablate its parameters in each phase concerning channel widths at each feature stride and target strides

N_k	c size	MR^O_{-2}	MAC (G)	Runtime (ms)
1	M	10.16	217.9	68
2	M	8.32	429.3	80
3	S	8.62	255.3	74
3	M	8.01	321.3	86
3	L	8.33	429.3	115
4	M	8.68	355.9	97

Table 3.2: The performance with different parameters and numbers of phases under the Caltech reasonable MR^O_{-2} setting. We further detail the efficiency of each setting in terms of multiply-accumulate (MAC) and runtime on an NVIDIA 1080 Ti.

to de-encode. Our primary method of AR-RPN uses what we refer to as medium channel width settings of $\mathbf{c}_M = \{128,\ 256,\ 512\}$. We further denote small and large channel settings such that $\mathbf{c}_S = \{64,\ 128,\ 256\}$ and $\mathbf{c}_L = \{256,\ 512,\ 512\}$, then train our AR-RPN with other settings kept consistent. Surprisingly, the small and large channel widths function similarly but neither as well as the medium, which roughly follows the rules-of-thumb channel settings outlined in VGG-16 [141]. For instance, the $\mathbf{c}_L$ and $\mathbf{c}_S$ achieve 8.33% ($\uparrow 0.32\%$) and 8.62% ($\uparrow 0.61\%$) miss rate, as detailed in Table 3.2. This suggests a difficulty when over or under expanding channels compared to the c width of source feature maps in $\mathbf{C}^1$.

We further analyze the runtime complexity of the de-encoder modules under each proposed setting in Table 3.2. Overall, we observe that channel width settings have a large effect on both multiply-accumulate (MAC) and runtime efficiencies of the AR-RPN. Specifically, channel width settings of $\mathbf{c}_S$, $\mathbf{c}_M$, and $\mathbf{c}_L$ respectively slow down by 8%, 26%, and 69% compared to $N_k = 1$ baseline.

What is the effect of convolutional re-sampling? Unlike previous decoder-encoder works [78, 98, 106, 119, 134], our module combines its re-sampling and feature generation into single convolutional re-sampling layers using either stride of 2 or fractional $\frac{1}{2}$ strides. To better understand the importance of this combined operation, we split every convolutional re-sampling layer $e(\cdot)$ and

$d(\cdot)$ into 2 separate layers: a bilinear re-sampling layer and a convolution feature generation layer. We observe that this separation causes performance to degrade from $8.01\% \rightarrow 9.45\%$ miss rate. This degradation suggests that providing the network with more freedom in re-sampling, as opposed to fixing the kernels to bilinear (or nearest neighbor), is beneficial for detection. Moreover, separating the operations into 2-steps is naturally less efficient concerning memory usage and runtime. Specifically, using the proposed convolutional re-sampling layers within AR-RPN consumes 41% less GPU memory compared to using a 2-step bilinear / convolution process and maintains a 16% faster runtime speed at inference.

How many autoregressive phases to stack? The use of autoregressive phases is clearly a *critical* component of our framework. Therefore, to understand its impact we ablate our framework by varying the number of phases while keeping all other settings constant. We report the performance of each setting in Table 3.2. Unsurprisingly, as fewer phases are used the performance is steeply reduced. For instance, recall that our 3-stage method achieves 8.01% miss rate. By removing a single phase, the miss rate increases by $\uparrow 0.32\%$ while only gaining 6 ms in runtime efficiency. When another phase is removed, an extreme degradation of $\uparrow 2.15\%$ is observed. Hence, the effect of additional phases seems to diminish with N_k such that the first additional phase has the highest impact, as suggested by Fig. 3.4. We further add a $4th$ phase following the same trend in incremental labeling ($h_4 = 0.7$) and observe that the performance begins to worsen. We suspect using more dense anchor sampling may help train the very high IoU threshold.

How to choose incremental labeling policies? Labeling policies are an important component to our autoregressive framework. We demonstrate the level of sensitivity and importance when using a variety of incremental labeling policies. Since high value IoU labeling policies only admit *very* well localized boxes as foreground, we refer to the IoU labeling policy of $h \geq 0.4$ as lenient,

Labeling Policy	MR^{O}_{-2}
no autoregressive	9.06
strict $\rightarrow$ lenient	9.03
moderate $\rightarrow$ moderate	8.94
strict $\rightarrow$ strict	8.43
lenient $\rightarrow$ strict	8.01

Table 3.3: The effects of labeling policies on the Caltech dataset under the reasonable MR^{O}_{-2} setting.

$h \geq 0.5$ as moderate, $h \geq 0.6$ as strict. We train the AR-RPN using labeling techniques of strict-to-lenient, moderate-to-moderate, strict-to-strict, and our primary setting of lenient-to-strict, as shown in Table 3.3. The strict-to-lenient method performs the worse among all settings, degrading by 1.02% MR. The moderate-to-moderate performs similarly and degrades by 0.80% MR. As shown in Fig. 3.4, the primary labeling policy of lenient-to-strict enables the network to start with large clusters of pedestrian box detections and iteratively suppress, resulting in more tight and peaky prediction maps. In contrast, strict-to-strict does not *ease* this transition as well resulting in a degradation of 0.42% MR. We further validate the effect by analyzing the score distributions across **all** pedestrians in the X/Y directions for the Caltech test dataset, as shown in Fig. 3.5. We observe a consistent trend in both directions where each successive phase results in a sharper peak with respect to its mean score. Each other labeling policy encourages the opposite or encourages the *same* predictions but more accurately. On a related point, we furhter examine the disagreements between phases ($\Delta P_{1\rightarrow 3}$ colored magenta, Fig. 3.4) which re-affirms phases logically agree on centroids of pedestrians. This analysis further shows that most suppression appears to be due to poorly localized boxes primarily in Y-direction (e.g., offset from the legs or head of a pedestrian).

For completeness, we further evaluate the extreme case where there is **no** incremental supervision or autoregressive flow within the network as included in Table 3.3. In this case, the core 3-phase network architecture is kept intact, except the prediction layers and concatenation have

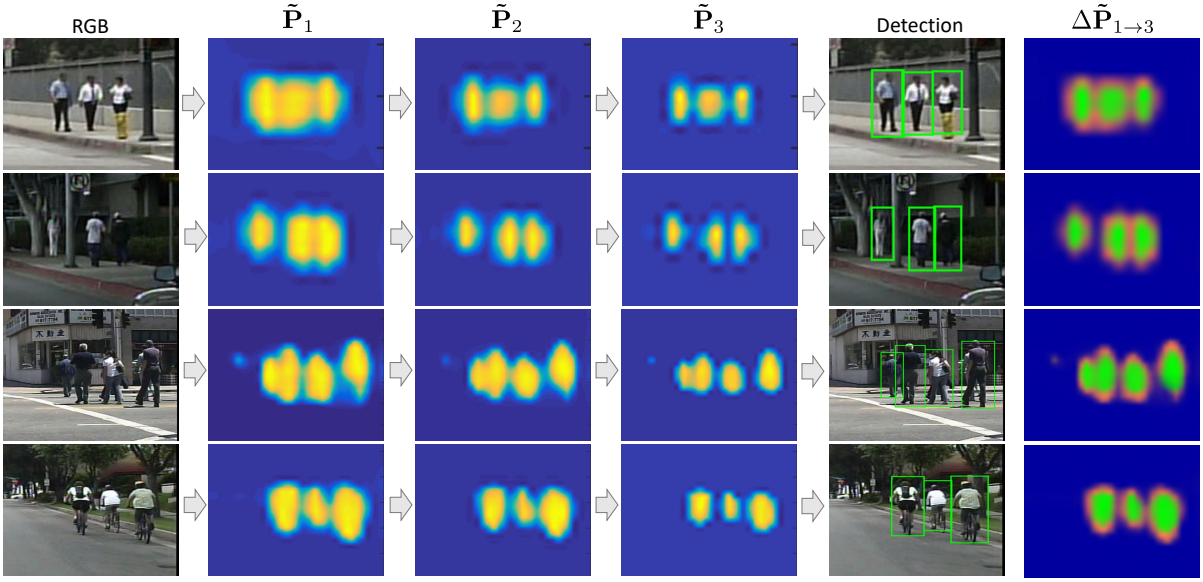

Figure 3.4: We visualize the prediction maps $\tilde{\mathbf{P}}_k$ of each phase by taking the maximum of foreground scores across all A anchors at each spatial location, i.e., denoting $\mathbf{P}_k = \{\mathbf{P}_k^{bg}, \mathbf{P}_k^{fg}\}$, we define $\tilde{\mathbf{P}}_k = \max_A \mathbf{P}_k^{fg}$. We use scaled blue $\to$yellow colors to visualize $\tilde{\mathbf{P}}_k$, where yellowness indicates high detection confidence. The detections of each phase become increasingly tighter and more adept to non-maximum suppression due to the incremental supervision for each phase (Sec. 3.3.2). We further analyze the prediction disagreements between phases $\Delta 1 \to 3$, shown in the right column, where green represents the agreement of the foreground and magenta the regions suppressed.

been removed from phases $1 \to 2$ and $2 \to 3$, therefore there is no incremental labeling policy to be decided. In doing so, the detection performance degrades by a considerable 2.14% miss rate, which furt

3.5 Summary

In this work, we present an autoregressive pedestrian detection framework which utilizes a novel stackable de-encoder module with convolutional re-sampling layers. The proposed AR-Ped framework is able to autoregressively produce and refine both features and classification predictions. In consequence, the collective phases approximate an ensemble of increasingly more precise classifi-

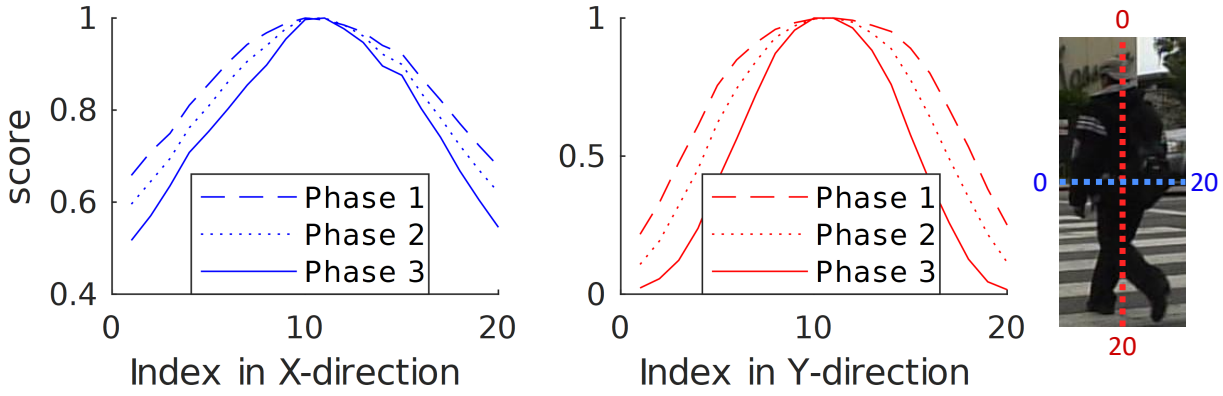

Figure 3.5: We analyze the mean prediction score ($\tilde{\mathbf{P}}_k$) of 20 uniformly sampled points along the center lines of X-direction (left) and Y-direction (right) averaged over **all** ground-truth pedestrians in Caltech test dataset, using bilinear interpolation when necessary. We note that successive phase scores form more *peaky* inclines radiating from the center of the pedestrian.

cation decisions and results in an overall improved classifier for pedestrian detection. We specifically supervise each phase using increasingly stricter labeling policies such that each phase of the network has similar recall as the last but with tighter and more clusterable prediction maps. We provide comprehensive ablation experiments to better understand and support each proposed component of our framework. We attain new state-of-the-art results on the Caltech dataset throughout many challenging experimental settings and achieve a highly competitive accuracy on the KITTI benchmark.

Chapter 4

M3D-RPN: Monocular 3D Region Proposal Network for Object Detection

Understanding the world in 3D is a critical component of urban autonomous driving. Generally, the combination of expensive LiDAR sensors and stereo RGB imaging has been paramount for successful 3D object detection algorithms, whereas monocular image-only methods experience drastically reduced performance. We propose to reduce the gap by reformulating the monocular 3D detection problem as a standalone 3D region proposal network. We leverage the geometric relationship of 2D and 3D perspectives, allowing 3D boxes to utilize well-known and powerful convolutional features generated in the image-space. To help address the strenuous 3D parameter estimations, we further design depth-aware convolutional layers which enable location specific feature development and in consequence improved 3D scene understanding. Compared to prior work in monocular 3D detection, our method consists of only the proposed 3D region proposal network rather than relying on external networks, data, or multiple stages. M3D-RPN is able to significantly improve the performance of both monocular 3D Object Detection and Bird's Eye

View tasks within the KITTI urban autonomous driving dataset, while efficiently using a shared multi-class model.

4.1 Introduction

Scene understanding in 3D plays a principal role in designing effective real-world systems such as in urban autonomous driving [5, 27, 51] and robotics [60, 151]. Currently, the foremost methods [30,95,126,138,174] on 3D detection rely extensively on expensive LiDAR sensors to provide sparse depth data as input. In comparison, monocular image-only 3D detection [23, 24, 118, 172] is considerably more difficult due to an inherent lack of depth cues. As a consequence, the performance gap between LiDAR-based methods and monocular approaches remains substantial.

Prior work on monocular 3D detection have each relied heavily on external state-of-the-art (SOTA) sub-networks, which are individually responsible for performing point cloud generation [24], semantic segmentation [23], 2D detection [118], or depth estimation [172]. A downside to such approaches is an inherent disconnection in component learning as well as system complexity. Moreover, reliance on additional sub-networks can introduce persistent noise, contributing to a limited upper-bound for the framework.

In contrast, we propose a single end-to-end region proposal network for multi-class 3D object detection (Fig. 4.1). We observe that 2D object detection performs reasonably and continues to make rapid advances [13,15,28,50,76,131]. The 2D and 3D detection tasks each aim to ultimately classify all instances of an object; whereas they differ in the dimensionality of their localization targets. Intuitively, we expect the power of 2D detection can be leveraged to guide and improve the performance of 3D detection, ideally within a unified framework rather than as separate components. Hence, we propose to reformulate the 3D detection problem such that both 2D and 3D

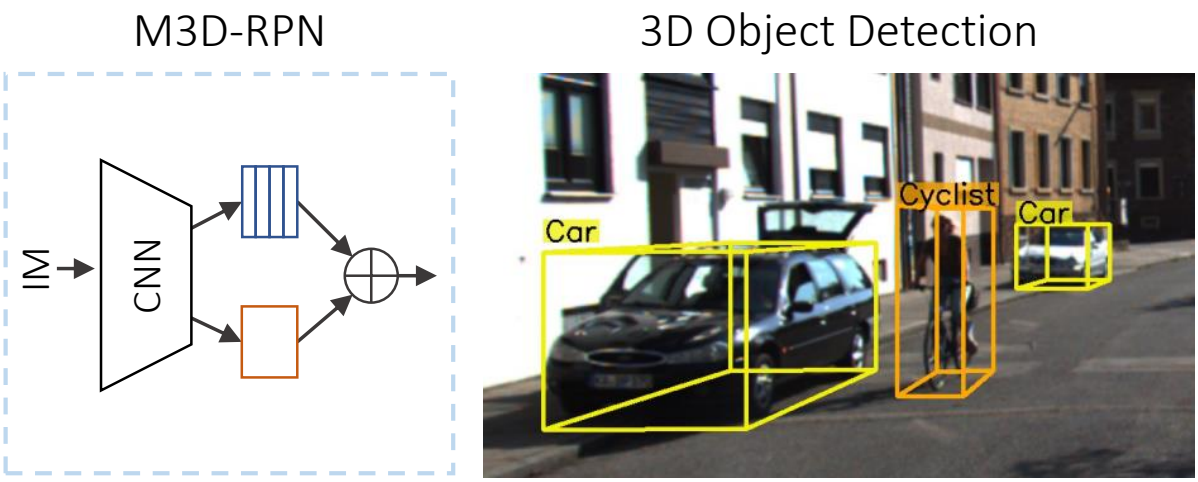

Figure 4.1: M3D-RPN uses a *single* monocular 3D region proposal network with global convolution (orange) and local depth-aware convolution (blue) to predict multi-class 3D bounding boxes.

spaces utilize shared anchors and classification targets. In doing so, the 3D detector is naturally able to perform on par with the performance of its 2D counterpart, from the perspective of reliably classifying objects. Therefore, the remaining challenge is reduced to 3D localization within the camera coordinate space.

To address the remaining difficultly, we propose three key designs tailored to improve 3D estimation. Firstly, we formulate 3D anchors to function primarily within the image-space and initialize all anchors with prior statistics for each of its 3D parameters. Hence, each discretized anchor inherently has a strong prior for reasoning in 3D, based on the consistency of a fixed camera viewpoint and the correlation between 2D scale and 3D depth. Secondly, we design a novel depth-aware convolutional layer which is able to learn spatially-aware features. Traditionally, convolutional operations are preferred to be spatially-invariant [79, 89] in order to detect objects at arbitrary image locations. However, while it is likely beneficial for low-level features, we show that high-level features improve when given increased awareness of their depth and while assuming a consistent camera scene geometry. Lastly, we optimize the orientation estimation θ using

3D $\rightarrow$ 2D projection consistency loss within a post-optimization algorithm. Hence, helping correct anomalies within θ estimation while assuming a reliable 2D bounding box.

To summarize, our contributions are the following:

- We formulate a standalone monocular 3D region proposal network (M3D-RPN) with a shared 2D and 3D detection space, while using prior statistics to serve as strong initialization for each 3D parameter.
- We propose depth-aware convolution to improve the 3D parameter estimation, thereby enabling the network to learn more spatially-aware high-level features.
- We propose a simple orientation estimation post-optimization algorithm which uses 3D projections and 2D detections to improve the θ estimation.
- We achieve state-of-the-art performance on the urban KITTI [51] benchmark for monocular Bird's Eye View and 3D Detection using a single multi-class network.

4.2 Related Work

2D Detection: Many works have addressed 2D detection in both generic [78,93,107,122,130] and urban scenes [10,11,13,15,108,109,131,175,189]. Most recent frameworks are based on seminal work of Faster R-CNN [132] due to the introduction of the region proposal network (RPN) as a highly effective method to efficiently generate object proposals. The RPN functions as a sliding window detector to check for the existence of objects at every spatial location of an image which match with a set of predefined template shapes, referred to as anchors. Despite that the RPN was conceived to be a preliminary stage within Faster R-CNN, it is often demonstrated to have promising effectiveness being extended to a single-shot standalone detector [107, 130, 166, 188]. Our framework builds upon the anchors of a RPN, specially designed to function in both the 2D

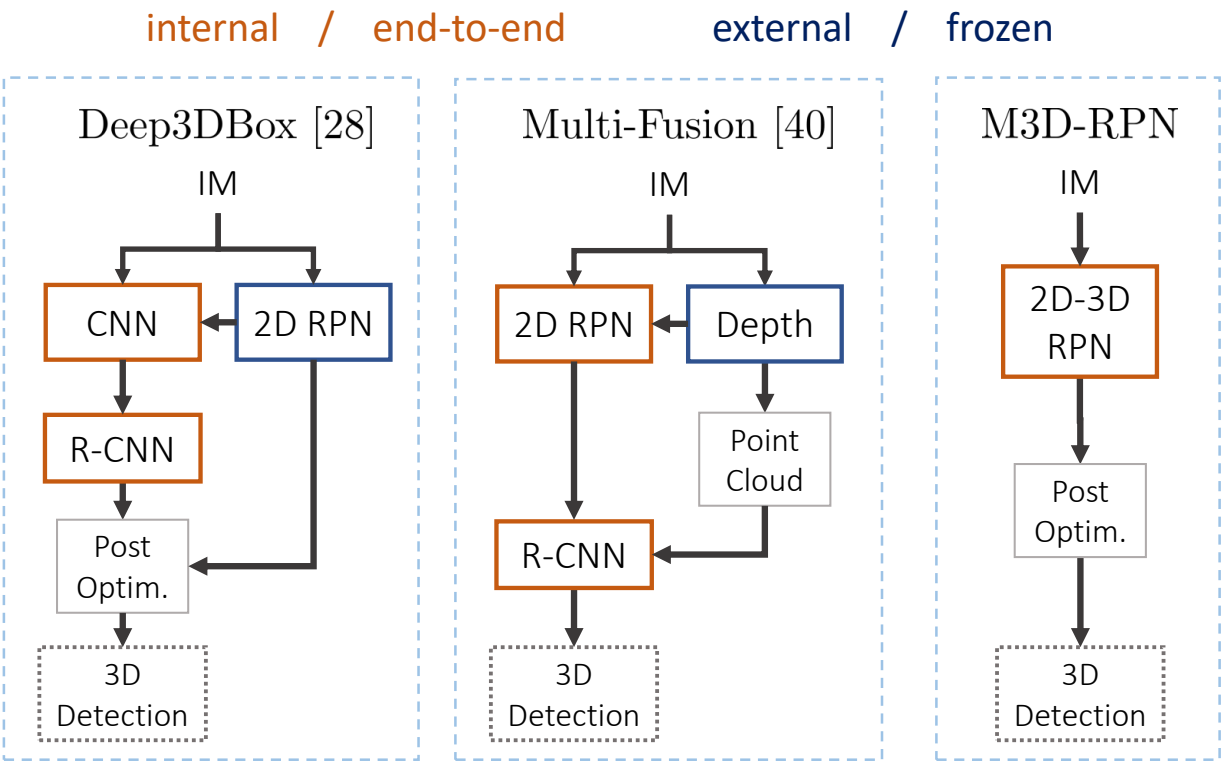

Figure 4.2: Comparison of Deep3DBox [118] and Multi-Fusion [172] with M3D-RPN. Notice that prior works are comprised of multiple internal stages (orange), and external networks (blue), whereas M3D-RPN is a *single-shot* network trained end-to-end.

and 3D spaces, and acting as a single-shot multi-class 3D detector.

LiDAR 3D Detection: The use of LiDAR data has proven to be essential input for SOTA frameworks [25, 30, 40, 95, 126, 138, 174] for 3D object detection applied to urban scenes. Leading methods tend to process sparse point clouds from LiDAR points [126, 138, 174] or project the point clouds into sets of 2D planes [25, 30]. While the LiDAR-based methods are generally high performing for a variety of 3D tasks, each is contingent on the availability of depth information generated from the LiDAR points or directly processed through point clouds. Hence, the methods are not applicable to camera-only applications as is the main purpose of our monocular 3D detection algorithm.

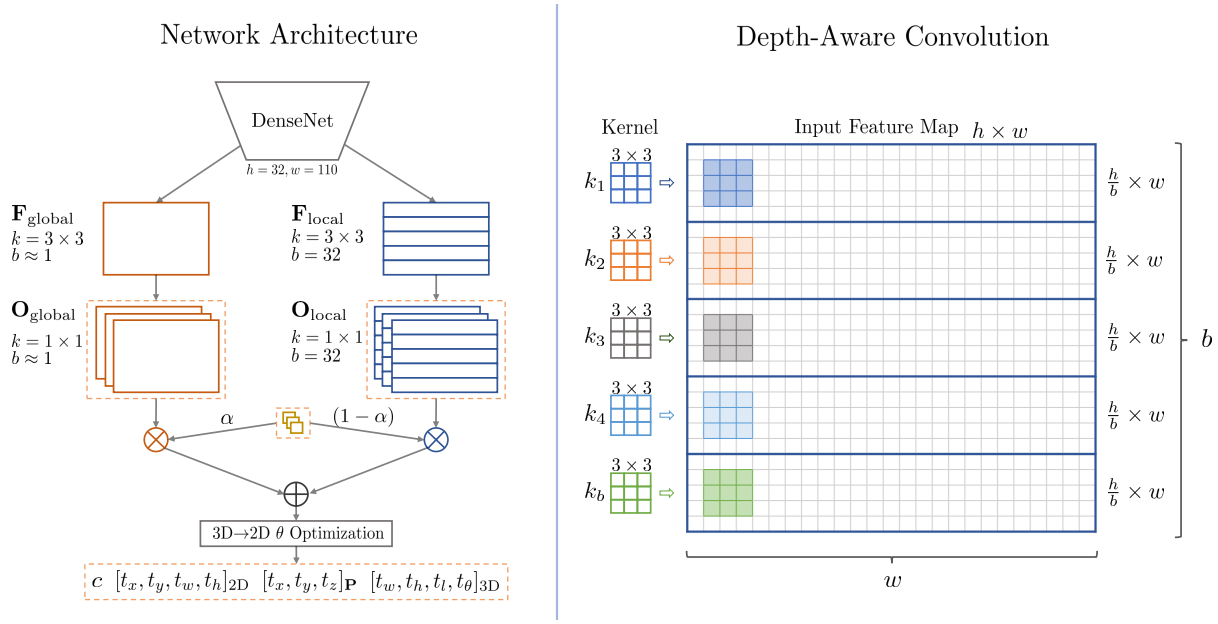

Figure 4.3: **Overview of M3D-RPN**. The proposed method consist of parallel paths for global (orange) and local (blue) feature extraction. The global features use regular spatial-invariant convolution, while the local features denote depth-aware convolution, as detailed right. The depth-aware convolution uses non-shared kernels in the row-space k_i for $i = 1 \dots b$, where b denotes the total number of distinct bins. To leverage both variants of features, we weightedly combine each output parameter from the parallel paths.

Image-only 3D Detection: 3D detection using only image data is inherently challenging due to an overall lack of reliable depth information. A common theme among SOTA image-based 3D detection methods [3, 23, 24, 118, 172] is to use a series of sub-networks to aid in detection. For instance, [24] uses a SOTA depth prediction with stereo processing to estimate point clouds. Then 3D cuboids are exhaustively placed along the ground plane given a known camera projection matrix, and scored based upon the density of the cuboid region within the approximated point cloud. As a follow-up, [23] adjusts the design from stereo to monocular by replacing the point cloud density heuristic with a combination of estimated semantic segmentation, instance segmentation, location, spatial context and shape priors, used while exhaustively classifying proposals on the ground plane.

In recent work, [118] uses an external SOTA object detector to generate 2D proposals then processes the cropped proposals within a deep neural network to estimate 3D dimensions and orientation. Similar to our work, the relationship between 2D boxes and 3D boxes projected onto the image plane is then exploited in post-processing to solve for the 3D parameters. However, our model directly predicts 3D parameters and thus only optimizes to improve θ, which converges in ~ 8 iterations in practice compared with 64 iterations in [118]. Xu *et al.* [172] utilize an additional network to predict a depth map which is subsequently used to estimate a LiDAR-like point cloud. The point clouds are then sampled using 2D bounding boxes generated from a separate 2D RPN. Lastly, a R-CNN classifier receives an input vector consisting of the sampled point clouds and image features, to estimate the 3D box parameters.

In contrast to prior work, we propose a *single* network trained only with 3D boxes, as opposed to using a set of external networks, data sources, and composed of multiple stages. Each prior work [23, 24, 118, 172] use external networks for at least one component of their framework, some of which have also been trained on external data. To the best of our knowledge, our method is the first to generate 2D and 3D object proposals simultaneously using a Monocular 3D Region Proposal Network (M3D-RPN). In theory, M3D-RPN is complementary to prior work and may be used to replace the proposal generation stage. A comparison between our method and prior is further detailed in Fig. 4.2.

4.3 M3D-RPN

Our framework is comprised of three key components. First, we detail the overall formulation of our multi-class 3D region proposal network. We then outline the details of depth-aware convolution and our collective network architecture. Finally, we detail a simple, but effective, post-optimization

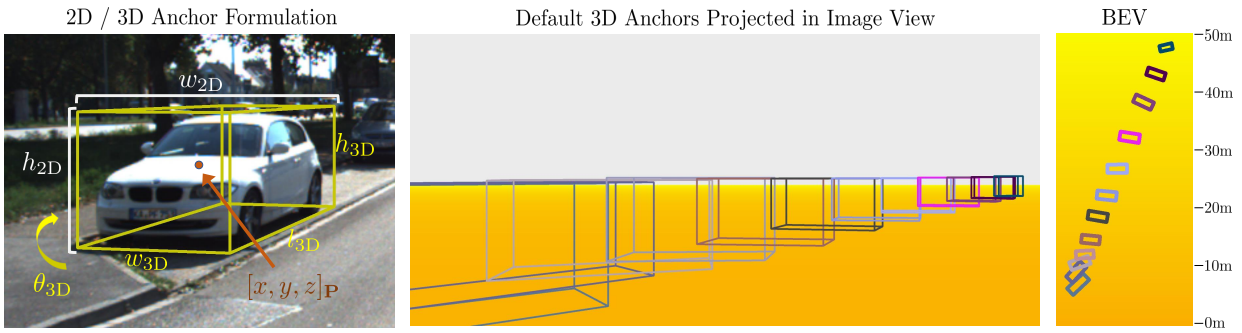

Figure 4.4: **Anchor Formulation** and **Visualized 3D Anchors**. We depict each parameter of within the 2D / 3D anchor formulation (left). We visualize the precomputed 3D priors when 12 anchors are used after projection in the image view (middle) and Bird's Eye View (right). For visualization purposes only, we span anchors in specific x_{3D} locations which best minimize overlap when viewed.

algorithm for increased 3D→2D consistency. We refer to our method as Monocular 3D Region Proposal Network (M3D-RPN), as illustrated in Fig. 4.3.

4.3.1 Formulation

The core foundation of our proposed framework is based upon the principles of the region proposal network (RPN) first proposed in Faster R-CNN [132], tailored for 3D. From a high-level, the region proposal network acts as sliding window detector which scans every spatial location of an input image for objects matching a set of predefined anchor templates. Then matches are regressed from the discretized anchors into continuous parameters of the estimated object.

Anchor Definition: To simultaneously predict both the 2D and 3D boxes, each anchor template is defined using parameters of both spaces: $[w, h]_{2D}$, $z_{\mathbf{P}}$, and $[w, h, l, \theta]_{3D}$. For placing an anchor and defining the full 2D / 3D box, a shared center pixel location $[x, y]_{\mathbf{P}}$ must be specified. The parameters denoted as 2D are used as provided in pixel coordinates. We encode the depth parameter $z_{\mathbf{P}}$ by projecting the 3D center location $[x, y, z]_{3D}$ in camera coordinates into the image given a known projection matrix $\mathbf{P} \in \mathbb{R}^{3\times4}$ as

$$\begin{bmatrix} x \cdot z \\ y \cdot z \\ z \end{bmatrix}_{\mathrm{P}} = \mathbf{P} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{\mathrm{3D}} . \tag{4.1}$$

The θ_{3D} represents the observation viewing angle [51]. Compared to the Y-axis rotation in the camera coordinate system, the observation angle accounts for the relative orientation of the object with respect to the camera viewing angle rather than the Bird's Eye View (BEV) of the ground plane. Therefore, the viewing angle is intuitively more meaningful to estimate when dealing with image features. We encode the remaining 3D dimensions $[w, h, l]_{\mathrm{3D}}$ as given in the camera coordinate system.

The mean statistic for each $z_{\mathbf{P}}$ and $[w, h, l, \theta]_{\mathrm{3D}}$ is pre-computed for each anchor individually, which acts as strong prior to ease the difficultly in estimating 3D parameters. Specifically, for each anchor we use the statistics across all matching ground truths which have ≥ 0.5 intersection over union (IoU) with the bounding box of the corresponding $[w, h]_{\mathrm{2D}}$ anchor. As a result, the anchors represent discretized templates where the 3D priors can be leveraged as a strong initial guess, thereby assuming a reasonably consistent scene geometry. We visualize the anchor formulation as well as precomputed 3D priors in Fig. 4.4.

3D Detection: Our model predicts output feature maps per anchor for c, $[t_x, t_y, t_w, t_h]_{\mathrm{2D}}$, $[t_x, t_y, t_z]_{\mathbf{P}}$, $[t_w, t_h, t_l, t_\theta]_{\mathrm{3D}}$. Let us denote n_a the number of anchors, n_c the number of classes, and $h \times w$ the feature map resolution. As such, the total number of box outputs is denoted $n_b = w \times h \times n_a$, spanned at each pixel location $[x, y]_{\mathbf{P}} \in \mathbb{R}^{w \times h}$ per anchor. The first output c represents the shared classification prediction of size $n_a \times n_c \times h \times w$, whereas each other output has size $n_a \times h \times w$. The outputs of $[t_x, t_y, t_w, t_h]_{\mathrm{2D}}$ represent the 2D bounding box transformation, which we collectively

refer to as b_{2D}. Following [132], the bounding box transformation is applied to an anchor with $[w, h]_{\text{2D}}$ as:

$$\begin{aligned} x'_{\text{2D}} &= x_{\mathbf{P}} + t_{x_{\text{2D}}} \cdot w_{\text{2D}}, \qquad & w'_{\text{2D}} &= \exp(t_{w_{\text{2D}}}) \cdot w_{\text{2D}}, \\ y'_{\text{2D}} &= y_{\mathbf{P}} + t_{y_{\text{2D}}} \cdot h_{\text{2D}}, \qquad & h'_{\text{2D}} &= \exp(t_{h_{\text{2D}}}) \cdot h_{\text{2D}}, \end{aligned} \tag{4.2}$$

where $x_{\mathbf{P}}$ and $y_{\mathbf{P}}$ denote spatial center location of each box. The transformed box b'_{2D} is thus defined as $[x, y, w, h]'_{\text{2D}}$. The following 7 outputs represent transformations denoting the projected center $[t_x, t_y, t_z]_{\mathbf{P}}$, dimensions $[t_w, t_h, t_l]_{\text{3D}}$ and orientation $t_{\theta_{\text{3D}}}$, which we collectively refer to as b_{3D}. Similar to 2D, the transformation is applied to an anchor with parameters $[w, h]_{\text{2D}}$, $z_{\mathbf{P}}$, and $[w, h, l, \theta]_{\text{3D}}$ as follows:

$$\begin{aligned} x'_{\mathbf{P}} &= x_{\mathbf{P}} + t_{x_{\mathbf{P}}} \cdot w_{\text{2D}}, \qquad & w'_{\text{3D}} &= \exp(t_{w_{\text{3D}}}) \cdot w_{\text{3D}}, \\ y'_{\mathbf{P}} &= y_{\mathbf{P}} + t_{y_{\mathbf{P}}} \cdot h_{\text{2D}}, \qquad & h'_{\text{3D}} &= \exp(t_{h_{\text{3D}}}) \cdot h_{\text{3D}}, \\ z'_{\mathbf{P}} &= t_{z_{\mathbf{P}}} + z_{\mathbf{P}}, \qquad & l'_{\text{3D}} &= \exp(t_{l_{\text{3D}}}) \cdot l_{\text{3D}}, \\ \theta'_{\text{3D}} &= t_{\theta_{\text{3D}}} + \theta_{\text{3D}}. \end{aligned} \tag{4.3}$$

Hence, b'_{3D} is then denoted as $[x, y, z]'_{\mathbf{P}}$ and $[w, h, l, \theta]'_{\text{3D}}$. As described, we estimate the projected 3D center rather than camera coordinates to better cope with the convolutional features based exclusively in the image space. Therefore, during inference we back-project the projected 3D center location from the image space $[x, y, z]'_{\mathbf{P}}$ to camera coordinates $[x, y, z]'_{\text{3D}}$ by using the inverse of Eqn. 4.1.

Loss Definition: The network loss of our framework is formed as a multi-task learning problem composed of classification L_c and a box regression loss for 2D and 3D, respectfully denoted as $L_{b_{\text{2D}}}$ and $L_{b_{\text{3D}}}$. For each generated box, we check if there exists a ground truth with at least ≥ 0.5

IoU, as in [132]. If yes then we use the best matched ground truth for each generated box to define a target with τ class index, 2D box $\hat{b}_{\text{2D}}$, and 3D box $\hat{b}_{\text{3D}}$. Otherwise, τ is assigned to the catch-all background class and bounding box regression is ignored. A softmax-based multinomial logistic loss is used to supervise for L_{c} defined as:

$$L_{\text{c}} = -\log\left(\frac{\exp(c_\tau)}{\Sigma_i^{nc}\exp(c_i)}\right). \tag{4.4}$$

We use a negative logistic loss applied to the IoU between the matched ground truth box $\hat{b}_{\text{2D}}$ and the transformed b'_{2D} for $L_{b_{\text{2D}}}$, similar to [160, 179], defined as:

$$L_{b_{\text{2D}}} = -\log\left(\text{IoU}(b'_{\text{2D}}, \hat{b}_{\text{2D}})\right). \tag{4.5}$$

The remaining 3D bounding box parameters are each optimized using a Smooth L_1 [54] regression loss applied to the transformations b_{3D} and the ground truth transformations $\hat{g}_{\text{3D}}$ (generated using $\hat{b}_{\text{3D}}$ following the inverse of Eqn. 4.3):

$$L_{b_{\text{3D}}} = \text{Smooth}L_1(b_{\text{3D}}, \hat{g}_{\text{3D}}). \tag{4.6}$$

Hence, the overall multi-task network loss L, including regularization weights λ_1 and λ_2, is denoted as:

$$L = L_{\text{c}} + \lambda_1 L_{b_{\text{2D}}} + \lambda_2 L_{b_{\text{3D}}}. \tag{4.7}$$

4.3.2 Depth-aware Convolution

Spatial-invariant convolution has been a principal operation for deep neural networks in computer vision [79, 89]. We expect that low-level features in the early layers of a network can reasonably

be shared and are otherwise invariant to depth or object scale. However, we intuitively expect that high-level features related to 3D scene understanding are dependent on depth when a fixed camera view is assumed. As such, we propose depth-aware convolution as a means to improve the spatial-awareness of high-level features within the region proposal network, as illustrated in Fig. 4.3.

The depth-aware convolution layer can be loosely summarized as regular 2D convolution where a set of discretized depths are able to learn non-shared weights and features. We introduce a hyperparameter b denoting the number of row-wise bins to separate a feature map into, where each learns a unique kernel k. In effect, depth-aware kernels enable the network to develop location specific features and biases for each bin region, ideally to exploit the geometric consistency of a fixed viewpoint within urban scenes. For instance, high-level semantic features, such as encoding a feature for a large wheel to detect a car, are valuable at close depths but not generally at far depths. Similarly, we intuitively expect features related to 3D scene understanding are inherently related to their row-wise image position.

An obvious drawback to using depth-aware convolution is the increase of memory footprint for a given layer by $\times b$. However, the total theoretical FLOPS to perform convolution remains consistent regardless of whether kernels are shared. We implement the depth-aware convolution layer in PyTorch [125] by unfolding a layer L into b padded bins then re-purposing the group convolution operation to perform efficient parallel operations on a GPU[1].

[1] In practice, we observe a $10 - 20\%$ overhead for reshaping when implemented with parallel group convolution in PyTorch [125].

	Type	IoU ≥ 0.7 [val1 / val2 / test]			IoU ≥ 0.5 [val1 / val2]		
		Easy	Mod	Hard	Easy	Mod	Hard
Mono3D [23]	Mono	5.22 / - / -	5.19 / - / -	4.13 / - / -	30.50 / -	22.39 / -	19.16 / -
3DOP [24]	Stereo	12.63 / - / -	9.49 / - / -	7.59 / - / -	55.04 / -	41.25 / -	34.55 / -
Deep3DBox [118]	Mono	- / 9.99 / -	- / 7.71 / -	- / 5.30 / -	- / 30.02	- / 23.77	- / 18.83
Multi-Fusion [172]	Mono	22.03 / 19.20 / 13.73	13.63 / 12.17 / 9.62	11.60 / 10.89 / 8.22	55.02 / 54.18	36.73 / 38.06	31.27 / 31.46
M3D-RPN	Mono	25.94 / 26.86 / 26.43	21.18 / 21.15 / 18.36	17.90 / 17.14 / 16.24	55.37 / 55.87	42.49 / 41.36	35.29 / 34.08

Table 4.1: **Bird's Eye View**. Comparison of our method to image-only 3D localization frameworks on the Bird's Eye View task (AP_{BEV}).

	Type	IoU ≥ 0.7 [val1 / val2 / test]			IoU ≥ 0.5 [val1 / val2]		
		Easy	Mod	Hard	Easy	Mod	Hard
Mono3D [23]	Mono	2.53 / - / -	2.31 / - / -	2.31 / - / -	25.19 / -	18.20 / -	15.52 / -
3DOP [24]	Stereo	6.55 / - / -	5.07 / - / -	4.10 / - / -	46.04 / -	34.63 / -	30.09 / -
Deep3DBox [118]	Mono	- / 5.85 / -	- / 4.10 / -	- / 3.84 / -	- / 27.04	- / 20.55	- / 15.88
Multi-Fusion [172]	Mono	10.53 / 7.85 / 7.08	5.69 / 5.39 / 5.18	5.39 / 4.73 / 4.68	47.88 / 45.57	29.48 / 30.03	26.44 / 23.95
M3D-RPN	Mono	20.27 / 20.40 / 20.65	17.06 / 16.48 / 15.70	15.21 / 13.34 / 13.32	48.96 / 49.89	39.57 / 36.14	33.01 / 28.98

Table 4.2: **3D Detection**. Comparison of our method to image-only 3D localization frameworks on the 3D Detection task (AP_{3D}).

4.3.3 Network Architecture

The backbone of our network uses DenseNet-121 [67]. We remove the final pooling layer to keep the network stride at 16, then dilate each convolutional layer in the last DenseBlock by a factor of 2 to obtain a greater field-of-view.

We connect two parallel paths at the end of the backbone network. The first path uses regular convolution where kernels are shared spatially, which we refer to as global. The second path exclusively uses depth-aware convolution and is referred to as local. For each path, we append a proposal feature extraction layer using its respective convolution operation to generate $\mathbf{F}_{\text{global}}$ and $\mathbf{F}_{\text{local}}$. Each feature extraction layer generates 512 features using a 3×3 kernel with 1 padding and is followed by a ReLU non-linear activation. We then connect the 12 outputs to each $\mathbf{F}$ corresponding to $c, [t_x, t_y, t_w, t_h]_{\text{2D}}, [t_x, t_y, t_z]_{\mathbf{P}}, [t_w, t_h, t_l, t_\theta]_{\text{3D}}$. Each output uses a 1×1 kernel and are collectively denoted as $\mathbf{O}_{\text{global}}$ and $\mathbf{O}_{\text{local}}$. To leverage the depth-aware and spatial-invariant strengths, we fuse each output using a learned attention α (after sigmoid) applied for $i = 1 \dots 12$

Algorithm 1 Post 3D→2D Algorithm. The algorithm takes input of 2D / 3D box b'_{2D}, $[x, y, z]'_{\mathbf{P}}$, $[w, h, l, \theta]'_{\text{3D}}$, step size σ, termination β, and decay γ parameters, then iteratively tunes θ via L_1 corner consistency loss.

Input: $b'_{\text{2D}}, [x, y, z]'_{\mathbf{P}}, [w, h, l, \theta]'_{\text{3D}}, \sigma, \beta, \gamma$
$\rho \leftarrow$ box-project$([x, y, z]_{\mathbf{P}}, [w, h, l, \theta - \sigma]_{\text{3D}})$
$\eta \leftarrow L_1(b'_{\text{2D}},\ \rho)$
while $\sigma \geq \beta$ **do**
  $\rho^- \leftarrow$ box-project$([x, y, z]_{\mathbf{P}}, [w, h, l, \theta - \sigma]_{\text{3D}})$
  $\rho^+ \leftarrow$ box-project$([x, y, z]_{\mathbf{P}}, [w, h, l, \theta + \sigma]_{\text{3D}})$
  $loss^- \leftarrow L_1(b'_{\text{2D}},\ \rho^-)$
  $loss^+ \leftarrow L_1(b'_{\text{2D}},\ \rho^+)$
  if $\min(loss^-, loss^+) > \eta$ **then**
    $\sigma \leftarrow \sigma \cdot \gamma$;
  else if $loss^- < loss^+$ **then**
    $\theta \leftarrow \theta - \sigma$;
    $\eta \leftarrow loss^-$
  else
    $\theta \leftarrow \theta + \sigma$;
    $\eta \leftarrow loss^+$
  end
end

as follows:

$$\mathbf{O}^i = \mathbf{O}^i_{\text{global}} \cdot \alpha_i + \mathbf{O}^i_{\text{local}} \cdot (1 - \alpha_i). \tag{4.8}$$

4.3.4 Post 3D→2D Optimization

We optimize the orientation parameter θ in a simple but effective post-processing algorithm (as detailed in Alg. 1). The proposed optimization algorithm takes as input both the 2D and 3D box estimations b'_{2D}, $[x, y, z]'_{\mathbf{P}}$, and $[w, h, l, \theta]'_{\text{3D}}$, as well as a step size σ, termination β, and decay γ parameters. The algorithm then iteratively steps through θ and compares the projected 3D boxes with b'_{2D} using a L_1 loss. The 3D→2D box-project function is defined as follows:

$$\Upsilon_0 = \begin{bmatrix} -l & l & l & l & l & -l & -l & -1 \\ -h & -h & h & h & -h & -h & h & h \\ -w & -w & -w & w & w & w & w & -w \end{bmatrix}_{\text{3D}}' \; / \, 2,$$

$$\Upsilon_{\text{3D}} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \\ 0 & 0 & 0 \end{bmatrix} \Upsilon_0 + \mathbf{P}^{-1} \begin{bmatrix} x \cdot z \\ y \cdot z \\ z \\ 1 \end{bmatrix}_{\mathbf{P}}', \tag{4.9}$$

$$\Upsilon_{\mathbf{P}} = \mathbf{P} \cdot \Upsilon_{\text{3D}}, \qquad \Upsilon_{\text{2D}} = \Upsilon_{\mathbf{P}} ./ \Upsilon_{\mathbf{P}}[\phi_z],$$

$$x_{\text{min}} = \min(\Upsilon_{\text{2D}}[\phi_x]), \quad y_{\text{min}} = \min(\Upsilon_{\text{2D}}[\phi_y]),$$

$$x_{\text{max}} = \max(\Upsilon_{\text{2D}}[\phi_x]), \quad y_{\text{max}} = \max(\Upsilon_{\text{2D}}[\phi_y]).$$

where $\mathbf{P}^{-1}$ is the inverse projection after padding $[0, 0, 0, 1]$, and ϕ denotes an index for axis $[x, y, z]$. We then use the projected box parameterized by $\rho = [x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}}]$ and the source b'_{2D} to compute a L_1 loss, which acts as the driving heuristic. When there is no improvement to the loss using $\theta \pm \sigma$, we decay the step by γ and repeat while $\sigma \geq \beta$.

4.3.5 Implementation Details

We implement our framework using PyTorch [125] and release the code at `http://cvlab.cse.msu.edu/project-m3d-rpn.html`. To prevent local features from overfitting on a subset of the image regions, we initialize the local path with pretrained global weights. In this case, each stage is trained for $50k$ iterations. We expect higher degrees of data augmentation or an iterative binning schedule, e.g., $b = 2^i$ from $i = 0 \ldots \log_2(b_{\text{final}})$, could enable more ease of training at the cost of more complex hyperparameters.

We use a learning rate of 0.004 with a poly decay rate using power 0.9, a batch size of 2, and weight decay of 0.9. We set $\lambda_1 = \lambda_2 = 1$. All images are scaled to a height of 512 pixels. As such, we use $b = 32$ bins for all depth-aware convolution layers. We use 12 anchor scales ranging from 30 to 400 pixels following the power function of $30 \cdot 1.265^i$ for $i = 0 \dots 11$ and aspect ratios of $[0.5, 1.0, 1.5]$ to define a total of 36 anchors for multi-class detection. The 3D anchor priors are learned using these templates with the training dataset as detailed in Sec. 4.3.1. We apply NMS on the box outputs in the 2D space using a IoU criteria of 0.4 and filter boxes with scores < 0.75. The 3D $\rightarrow$ 2D optimization uses settings of $\sigma = 0.3\pi$, $\beta = 0.01$, and $\gamma = 0.5$. Lastly, we perform random mirroring and online hard-negative mining by sampling the top 20% high loss boxes in each minibatch.

We note that M3D-RPN relies on 3D box annotations and a known projection matrix $\mathbf{P}$ per sequence. For extension to a dataset without these known, it may be necessary to predict the camera intrinsics and utilize weak supervision leveraging 3D-2D projection geometry as loss constraints.

4.4 Experiments

We evaluate our proposed framework on the challenging KITTI [51] dataset under two core 3D localization tasks: Bird's Eye View (BEV) and 3D Object Detection. We comprehensively compare our method on the official test dataset as well as two validation splits [24, 169], and perform analysis of the critical components which comprise our framework. We further visualize qualitative examples of M3D-RPN on multi-class 3D object detection in diverse scenes (Fig. 4.5).

	AP_{BEV} [val1 / val2 / test]	AP_{3D} [val1 / val2 / test]
Car	21.18 / 21.15 / 18.36	17.06 / 16.48 / 15.70
Pedestrian	11.60 / 11.44 / 11.35	11.28 / 11.30 / 10.54
Cyclist	10.13 / 9.09 / 1.29	10.01 / 9.09 / 1.03

Table 4.3: **Multi-class 3D Localization**. The performance of our method when applied as a multi-class 3D detection system using a single shared model. We evaluate using the mod setting on KITTI.

4.4.1 KITTI

The KITTI [51] dataset provides many widely used benchmarks for vision problems related to self-driving cars. Among them, the Bird's Eye View (BEV) and 3D Object Detection tasks are most relevant to evaluate 3D localization performance. The official dataset consists of $7{,}481$ training images and $7{,}518$ testing images with 2D and 3D annotations for car, pedestrian, and cyclist. For each task we report the Average Precision (AP) under 3 difficultly settings: easy, moderate and hard as detailed in [51]. Methods are further evaluated using different IoU criteria per class. We emphasize our results on the official settings of IoU ≥ 0.7 for cars and IoU ≥ 0.5 for pedestrians and cyclists.

We conduct experiments on three common data splits including val1 [24], val2 [169], and the official test split [51]. Each split contains data from non-overlapping sequences such that no data from an evaluated frame, or its neighbors, have been used for training. We focus our comparison to SOTA prior work which use image-only input. We primarily compare our methods using the car class, as has been the focus of prior work [23, 24, 118, 172]. However, we emphasize that our models are trained as a shared multi-class detection system and therefore also report the multi-class capability for monocular 3D detection, as detailed in Tab. 4.3.

Bird's Eye View: The Bird's Eye View task aims to perform object detection from the overhead viewpoint of the ground plane. Hence, all 3D boxes are first projected onto the ground plane then

	2D Detection [val1 / test]		
	Easy	Mod	Hard
Mono3D [23]	93.89 / 92.33	88.67 / 88.66	79.68 / 78.96
3DOP [24]	93.08 / 93.04	88.07 / 88.64	79.39 / 79.10
Deep3DBox [118]	- / 92.98	- / 89.04	- / 77.17
Multi-Fusion [172]	- / 90.43	- / 87.33	- / 76.78
M3D-RPN	90.24 / 84.34	83.67 / 83.78	67.69 / 67.85

Table 4.4: **2D Detection**. The performance of our method evaluated on 2D detection using the car class on val1 and test datasets.

top-down 2D detection is applied. We evaluate M3D-RPN on each split as detailed in Tab. 4.1.

M3D-RPN achieves a notable improvement over SOTA image-only detectors across all data splits and protocol settings. For instance, under criteria of IoU ≥ 0.7 with val1, our method achieves 21.18% ($\uparrow 7.55\%$) on moderate, and 17.90% ($\uparrow 6.30\%$) on hard. We further emphasize our performance on test which achieves 18.36% ($\uparrow 8.74\%$) and 16.24% ($\uparrow 8.02\%$) respectively on moderate and hard settings with IoU ≥ 0.7, which is the most challenging setting.

3D Object Detection: The 3D object detection task aims to perform object detection directly in the camera coordinate system. Therefore, an additional dimension is introduced to all IoU computations, which substantially increases the localization difficulty compared to BEV task. We evaluate our method on 3D detection with each split under all commonly studied protocols as described in Tab. 4.2. Our method achieves a significant gain over state-of-the-art image-only methods throughout each protocol and split.

We emphasize that the current most difficult challenge to evalaute 3D localization is the 3D object detection task. Similarly, the moderate and hard settings with IoU ≥ 0.7 are the most difficult protocols to evaluate with. Using these settings with val1, our method notably achieves 17.06% ($\uparrow 11.37\%$) and 15.21 ($\uparrow 9.82\%$) respectively. We further observe similar gains on the other splits. For instance, when evaluated using the testing dataset, we achieve 15.70% ($\uparrow 10.52$)

and 13.32% ($\uparrow 8.64$) on the moderate and hard settings despite being trained as a *shared* multi-class model and compared to single model methods [23, 24, 118, 172]. When evaluated with less strict criteria such as IoU ≥ 0.5, our method demonstrates smaller but reasonable margins ($\sim 3-6\%$), implying that M3D-RPN has similar recall to prior art but significantly higher precision overall.

Multi-Class 3D Detection: To demonstrate generalization beyond a single class, we evaluate our proposed 3D detection framework on the car, pedestrian and cyclist classes. We conduct experiments on both the Bird's Eye View and 3D Detection tasks using the KITTI test dataset, as detailed in Tab. 4.3. Although there are not monocular 3D detection methods to compare with for multi-class, it is noteworthy that the performance on pedestrian outperforms prior work performance on *car*, which usually has the opposite relationship, thereby suggesting a reasonable performance. However, M3D-RPN is noticeably less stable for cyclists, suggesting a need for advanced sampling or data augmentation to overcome the data bias towards car and pedestrian.

2D Detection: We evaluate our performance on 2D car detection (detailed in Tab. 4.4). We note that M3D-RPN performs less compared to other 3D detection systems applied to the 2D task. However, we emphasize that prior work [23, 24, 118, 172] use external networks, data sources, and include multiple stages (e.g., Fast [54], Faster R-CNN [132]). In contrast, M3D-RPN performs all tasks simultaneously using only a single-shot 3D proposal network. Hence, the focus of our work is primarily to improve 3D detection proposals with an emphasis on the quality of 3D localization. Although M3D-RPN does not compete directly with SOTA methods for 2D detection, its performance is suitable to facilitate the tasks in focus such as BEV and 3D detection.

b	Post-Optim	AP_{2D}	AP_{3D}	AP_{BEV}	RT (ms)
		82.16	10.99	12.99	118
	✓	82.16	15.08	17.47	128
1	✓	82.88	12.87	17.91	133
4	✓	84.15	14.46	19.14	134
8	✓	83.86	16.04	20.99	143
16	✓	83.02	15.97	18.48	153
32	✓	83.67	17.06	21.18	161

Table 4.5: **Ablations**. We ablate the effects of b for depth-aware convolution and the post-optimization 3D→2D algorithm with respect to performance on moderate setting of cars and runtime (RT).

c	x_{2D}	y_{2D}	w_{2D}	h_{2D}	$x_{\mathbf{P}}$	$y_{\mathbf{P}}$	$z_{\mathbf{P}}$	w_{3D}	h_{3D}	l_{3D}	θ_{3D}	
33	48	47	45	45	44	45	44	42	38	43	38	%

Table 4.6: **Local and Global α weights**. We detail the α weights learned to individually fuse each global and local output. Lower implies higher weight towards the local depth-aware convolution.

4.4.2 Ablations

For all ablations and experimental analysis we use the KITTI val1 dataset split and evaluate utilizing the car class. Further, we use the moderate setting of each task which includes 2D detection, 3D detection, and BEV (Tab. 4.5).

Depth-aware Convolution: We propose depth-aware convolution as a method to improve the spatial-awareness of high-level features. To better understand the effect of depth-aware convolution, we ablate it from the perspective of the hyperparameter b which denotes the number of discrete bins. Since our framework uses an image scale of 512 pixels with network stride of 16, the output feature map can naturally be separated into $\frac{512}{16} = 32$ bins. We therefore ablate using bins of $[4, 8, 16, 32]$ as described in Tab. 4.5.

We additionally ablate the special case of $b = 1$, which is the equivalent to utilizing two global streams. We observe that both $b = 1$ and $b = 4$ result in generally worse performance than the baseline without local features, suggesting that arbitrarily adding deeper layers is not inherently

Figure 4.5: **Qualitative Examples**. We visualize qualitative examples of our method for multi-class 3D object detection. We use yellow to denote cars, green for pedestrians, and orange for cyclists. All illustrated images are from the val1 [24] split and not used for training.

helpful for 3D localization. However, we observe consistent improvements when $b = 32$ is used, achieving a large gain of 3.71% in AP_{BEV}, 1.98% in AP_{3D}, and 1.51% in AP_{2D}.

We breakdown the learned α weights after sigmoid which are used to fuse the global and local outputs (Tab. 4.6). Lower values favor local branch and vice-versa for global. Interestingly, the classification c output learns the highest bias toward local features, suggesting that semantic features in urban scenes have a moderate reliance on depth position.

Post 3D→2D Optimization: The post-optimization algorithm encourages consistency between 3D boxes projected into the image space and the predicted 2D boxes. We ablate the effectiveness of this optimization as detailed in Tab. 4.5. We observe that the post-optimization has a significant impact on both BEV and 3D detection performance. Specifically, we observe performance gains of 4.48% in AP_{BEV} and 4.09% in AP_{3D}. We additionally observe that the algorithm converges in approximately 8 iterations on average and adds minor 13 ms overhead (per image) to the runtime.

Efficiency: We emphasize that our approach uses only a single network for inference and hence

involves overall more direct 3D predictions than the use of multiple networks and stages (RPN with R-CNN) used in prior works [23,24,118,172]. We note that direct efficiency comparison is difficult due to a lack of reporting in prior work. However, we comprehensively report the efficiency of M3D-RPN for each ablation experiment, where b and post-optimization are the critical factors, as detailed in Tab. 4.5. The runtime efficiency is computed using NVIDIA 1080ti GPU averaged across the KITTI val1 dataset. We note that depth-aware convolution incurs $2-20\%$ overhead cost for $b = 1 \dots 32$, caused by unfolding and reshaping in PyTorch [125].

4.5 Summary

In this work, we present a reformulation of monocular image-only 3D object detection using a *single-shot* 3D RPN, in contrast to prior work which are comprised of external networks, data sources, and involve multiple stages. M3D-RPN is uniquely designed with shared 2D and 3D anchors which leverage strong priors closely linked to the correlation between 2D scale and 3D depth. To help improve 3D parameter estimation, we further propose depth-aware convolution layers which enable the network to develop spatially-aware features. Collectively, we are able to significantly improve the performance on the challenging KITTI dataset on both the Birds Eye View and 3D object detection tasks for the car, pedestrian, and cyclist classes.

Acknowledgment: Research was partially sponsored by the Army Research Office under Grant Number W911NF-18-1-0330. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Chapter 5

Kinematic 3D Object Detection in Monocular Video

Although temporal motion is an invaluable resource to human vision for detection, tracking, and depth perception, such features have not been thoroughly utilized in modern 3D object detectors. In this work, we propose a novel method for monocular video-based 3D object detection which leverages kinematic motion to extract scene dynamics and improve localization accuracy. We first propose a novel decomposition of object orientation and a self-balancing 3D confidence. We show that both components are critical to enable our kinematic model to work effectively. Collectively, using only a single model, we efficiently leverage 3D kinematics from monocular videos to improve the overall localization precision in 3D object detection while also producing useful by-products of scene dynamics (ego-motion and per-object velocity). We achieve state-of-the-art performance on monocular 3D object detection and the Bird's Eye View tasks within the KITTI self-driving dataset.

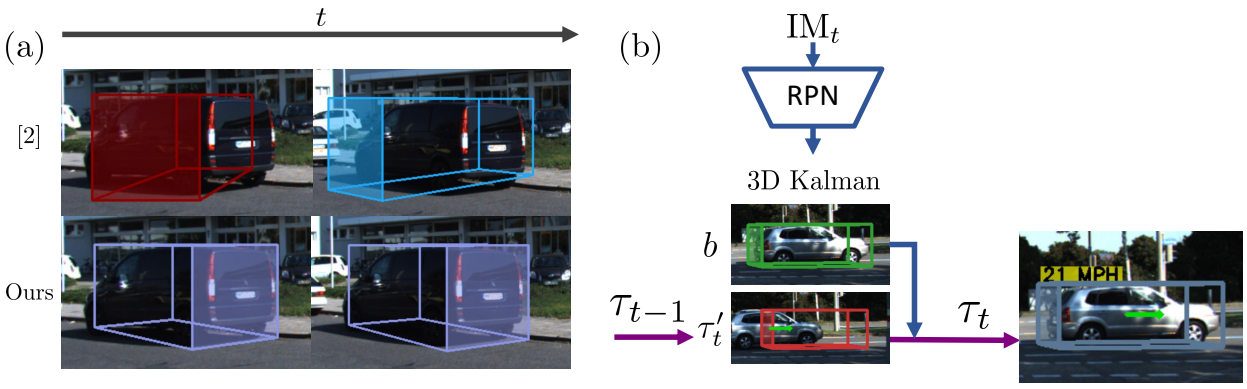

Figure 5.1: Single-frame 3D detection [9] often has unstable estimation through time (a), while our video-based method (b) is more robust by leveraging **kinematic motion** via a 3D Kalman Filter to fuse forecasted tracks τ'_t and measurements b into τ_t.

5.1 Introduction

The detection of foreground objects is among the most critical requirements to facilitate self-driving applications [11, 39]. Recently, 3D object detection has made significant progress [26, 88, 94, 95, 139, 177], even while using only a monocular camera [9, 81, 90, 103, 113, 114, 140, 162]. Such works primarily look at the problem from the perspective of *single* frames, ignoring useful temporal cues and constraints.

Computer vision cherishes inverse problems, *e.g.*, recovering the 3D physical motion of objects from monocular videos. Motion information such as object velocity in the metric space is highly desirable for the path planning of self-driving. However, single image-based 3D object detection can not directly estimate physical motion, without relying on additional tracking modules. Therefore, *video-based 3D object detection* would be a sensible choice to recover such motion information. Furthermore, without modeling the physical motion, image-based 3D object detectors are naturally more likely to suffer from erratic and unnatural changes through time in orientation and localization (as exemplified in Fig. 5.1(a)). Therefore, we aim to build a novel video-based 3D object detector which is able to provide *accurate* and *smooth* 3D object detection with per-object

velocity, while also prioritizing a *compact* and *efficient* model overall.

Yet, designing an effective video-based 3D object detector has challenges. Firstly, motion which occurs in real-world scenes can come from a variety of sources such as the camera atop of an autonomous vehicle or robot, and/or from the scene objects themselves — for which most of the safety-critical objects (car, pedestrian, cyclist [51]) are typically *dynamic*. Moreover, using video inherently involves an increase in data consumption which introduces practical challenges for training and/or inference including with memory or redundant processing.

To address such challenges, we propose a novel framework to integrate a 3D Kalman filter into a 3D detection system. We find Kalman is an ideal candidate for three critical reasons: (1) it allows for use of real-world motion models to serve as a strong prior on object dynamics, (2) it is inherently efficient due to its recursive nature and general absence of parameters, (3) the resultant behavior is explainable and provides useful by-products such as the object velocity.

Furthermore, we observe that objects predominantly move in the direction indicated by their orientation. Fortunately, the benefit of Kalman allows us to integrate this real-world constraint into the motion model as a compact scalar velocity. Such a constraint helps maintain the consistency of velocity over time and enables the Kalman motion forecasting and fusion to perform accurately.

However, a model restricted to only move in the direction of its orientation has an obvious flaw — what if the orientation itself is *inaccurate*? We therefore propose a novel reformulation of orientation in favor of accuracy and stability. We find that our orientation improves the 3D localization accuracy by a margin of 2.39% and reduces the orientation error by $\approx 20\%$, which collectively help enable the proposed Kalman to function more effectively.

A notorious challenge of using Kalman comes in the form of *uncertainty*, which is conventionally [164] assumed to be known and static, *e.g.*, from a sensor. However, 3D objects in video are intuitively dependent on more complex factors of image features and cannot necessarily be treated

like a sensor measurement. For a better understanding of 3D uncertainty, we propose a 3D self-balancing confidence loss. We show that our proposed confidence has higher correlation with the 3D localization performance compared to the typical classification probability, which is commonly used in detection [130, 132].

To complete the full understanding of the scene motion, we elect to estimate the ego-motion of the capturing camera itself. Hence, we further narrow the work of Kalman to account for only the *object's* motion. Collectively, our proposed framework is able to model important scene dynamics, both ego-motion and per-object velocity, and more precisely detect 3D objects in videos using a stabilized orientation and 3D confidence estimation. We demonstrate that our method achieves state-of-the-art (SOTA) performance on monocular 3D Object Detection and Bird's Eye View (BEV) tasks in the KITTI dataset [51].

In summary, our contributions are as follows:

- We propose a monocular video-based 3D object detector, leveraging realistic motion constraints with an integrated ego-motion and a 3D Kalman filter.
- We propose to reformulate orientation into axis, heading and offset along with a self-balancing 3D localization loss to facilitate the stability necessary for the proposed Kalman filter to perform more effectively.
- Overall, using only a *single model* our framework develops a comprehensive 3D scene understanding including object cuboids, orientation, velocity, object motion, uncertainty, and ego-motion, as detailed in Fig. 5.1 and 5.2.
- We achieve a new SOTA performance on monocular 3D object detection and BEV tasks using comprehensive metrics within the KITTI dataset.

5.2 Related Work

We first provide context of our novelties from the perspective of monocular 3D object detection (Sec. 5.2.1) with attention to orientation and uncertainty estimation. We next discuss and contrast with video-based object detection (Sec. 5.2.2).

5.2.1 Monocular 3D Object Detection

Monocular 3D object detection has made significant progress [9, 23, 24, 81, 90, 103, 113, 114, 118, 140, 161, 172]. Early methods such as [24] began by generating 3D object proposals along a ground plane using object priors and estimated point clouds, culminating in an energy minimization approach. [23, 81, 172] utilize additional domains of semantic segmentation, object priors, and estimated depth to improve the localization. Similarly, [113, 162] create a pseudo-LiDAR map using SOTA depth estimator [20, 46, 49], which is respectfully passed into detection subnetworks or LiDAR-based 3D object detection works [80, 126, 127]. In [90, 103, 114, 118, 140] strong 2D detection systems are extended to add cues such as object orientation, then the remaining 3D box parameters are solved via 3D box geometry. [9] extends the region proposal network (RPN) of Faster R-CNN [132] with 3D box parameters.

5.2.1.1 Orientation Estimation:

Prior monocular 3D object detectors estimate orientation via two main techniques. The first method is to classify orientation via a series of discrete bins then regress a relative offset [23, 24, 81, 90, 103, 114, 118, 172]. The bin technique requires a trade-off between the quantity/coverage of the discretized angles and an increase in the number of estimated parameters (bin $\times$). Other methods directly regress the orientation angle using quaternion [140] or Euler [9, 113] angles. Direct

regression is comparatively efficient, but may lead to degraded performance and periodicity challenges [190], as exemplified in Fig. 5.1.

In contrast, we propose a novel orientation decomposition which serves as an intuitive compromise between the bin and direct approaches. We decompose the orientation estimation into three components: axis and heading classification, followed by an angle offset. Thus, our technique increases the parameters by a *static* factor of 2 compared to a bin hyperparameter, while drastically reducing the offset search space for each orientation estimation (discussed in Sec. 5.3.1).

5.2.1.2 Uncertainty Estimation:

Although it is common to utilize the classification score to rate boxes in 2D object detection [18, 130, 132, 175] or explicitly model uncertainty as parametric estimation [84], prior works in monocular 3D object detection realize the need for 3D box uncertainty/confidence [103, 140]. [103] defines confidence using the 3D IoU of a box and ground truth after center alignment, thus capturing the confidence primarily of the 3D *object dimensions*. [140] predicts a confidence by re-mapping the 3D box loss into a probability range, which intuitively represents the confidence of the overall 3D box accuracy.

In contrast, our *self-balancing* confidence loss is generic and self-supervised, with two benefits. (1) It enables estimation of a 3D localization confidence using only the loss values, thus being more general than 3D IoU. (2) It enables the network to naturally re-balance extremely hard 3D boxes and focus on relatively achievable samples. Our ablation (Sec. 5.4.4) shows the importance of both effects.

5.2.2 Video-based Object Detection

Video-based object detection [6, 104, 171, 191, 192] is generally less studied than single-frame object detection [18, 130–132, 175, 177]. A common trend in video-based detection is to benefit the accuracy-efficiency trade-off via reducing the frame redundancy [6, 104, 171, 191, 192]. Such works are applied primarily on domains of ImageNet VID 2015 [135], which contain less ego-motion from the capturing camera than self-driving scenarios [31, 51]. As such, the methods are designed to use 2D transformations, which lack the consistency and realism of 3D motion modeling.

In comparison, to our knowledge this is the first work that *utilizes video cues to improve the accuracy and robustness of monocular 3D object detection*. In the domain of 2D/3D object tracking, [52] experiments using Kalman Filters, Particle Filters, and Gaussian Mixture Models, and observe Kalman to be the most effective aggregation method for tracking. An LSTM with depth ordering and IMU camera ego-motion is utilized in [66] to improve the tracking accuracy. In contrast, we explore how to naturally and effectively leverage a 3D Kalman filter to improve the accuracy and robustness of monocular 3D object detection. We propose novel enhancements including estimating ego-motion, orientation, and a 3D confidence, while efficiently using only a single model.

5.3 Methodology

Our proposed kinematic framework is composed of three primary components: a 3D region proposal network (RPN), ego-motion estimation, and a novel kinematic model to *take advantage of temporal motion in videos*. We first overview the foundations of a 3D RPN. Then we detail our contributions of orientation decomposition and self-balancing 3D confidence, which are integral

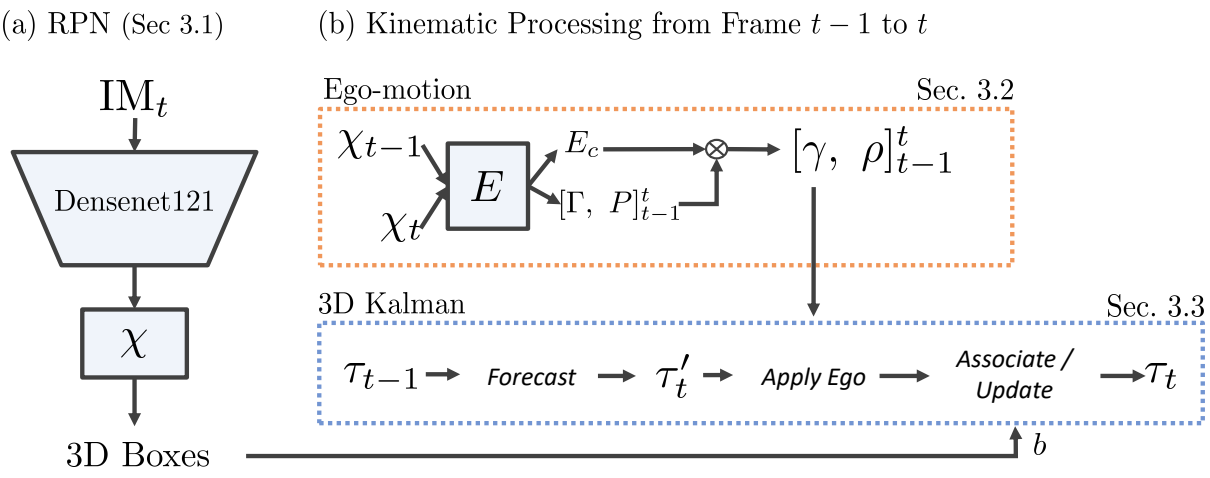

Figure 5.2: **Overview.** Our framework uses a RPN to first estimate 3D boxes (Sec. 5.3.1). We forecast previous frame tracks τ_{t-1} into τ'_t using the estimated Kalman velocity. Self-motion is compensated for applying a global ego-motion (Sec. 5.3.2) to tracks τ'_t. Lastly, we fuse τ'_t with measurements b using a kinematic 3D Kalman filter (Sec. 5.3.3).

to the kinematic method. Next we detail ego-motion estimation. Lastly, we present the complete kinematic framework (Fig. 5.2) which carefully employs a 3D Kalman [164] to model realistic motion using the aforementioned components, ultimately producing a more accurate and comprehensive 3D scene understanding.

5.3.1 Region Proposal Network

Our measurement model is founded on the 3D RPN [9], enhanced using novel orientation and confidence estimations. The RPN itself acts as a sliding window detector following the typical practices outlined in Faster R-CNN [132] and [9]. Specifically, the RPN consists of a backbone network and a detection head which predicts 3D box outputs relative to a set of predefined anchors.

5.3.1.1 Anchors:

We define our 2D-3D anchor Φ to consist of 2D dimensions $[\Phi_w, \Phi_h]_{\text{2D}}$ in pixels, a projected depth-buffer Φ_z in meters, 3D dimensions $[\Phi_w, \Phi_h, \Phi_l]_{\text{3D}}$ in meters, and orientations with respect to two major axes $[\Phi_0, \Phi_1]_{\text{3D}}$ in radians. The term Φ_z is related to the camera coordinate $[x, y, z]_{\text{3D}}$ by the equation $\Phi_z \cdot [u, v, 1]^{\text{T}}_{\text{2D}} = \boldsymbol{\Upsilon} \cdot [x, y, z, 1]^{\text{T}}_{\text{3D}}$ where $\boldsymbol{\Upsilon} \in \mathbb{R}^{3\times4}$ is a known projection matrix. We compute the anchor values by taking the mean of each parameter after clustering all ground truth objects in 2D, following the process in [9].

5.3.1.2 3D Box Outputs:

Since our network is based on the principles of a RPN [9, 132], most of the estimations are defined as a transformation $\mathbf{T}$ relative to an anchor. Let us define n_a as the number of anchors, n_c as the number of object classes, and $w \times h$ as the output resolution of the network. The RPN outputs a classification map $\mathbf{C} \in \mathbb{R}^{(n_a \cdot n_c)\times w\times h}$, then 2D transformations $[\mathbf{T}_x, \mathbf{T}_y, \mathbf{T}_w, \mathbf{T}_h]_{\text{2D}}$, 3D transformations $[\mathbf{T}_u, \mathbf{T}_v, \mathbf{T}_z, \mathbf{T}_w, \mathbf{T}_h, \mathbf{T}_l, \mathbf{T}_{\theta_r}]_{\text{3D}}$, axis and heading $[\boldsymbol{\Theta}_a, \boldsymbol{\Theta}_h]$, and lastly a 3D self-balancing confidence $\boldsymbol{\Omega}$. Each output has a size of $\mathbb{R}^{n_a\times w\times h}$. The outputs can be unrolled into $n_b = (n_a \cdot w \cdot h)$ boxes with $(n_c + 14)$-dim, with parameters of c, $[t_x, t_y, t_w, t_h]_{\text{2D}}$, $[t_u, t_v, t_z, t_w, t_h, t_l, t_{\theta_r}]_{\text{3D}}$, $[\theta_a, \theta_h]$, and ω, which relate to the maps by $c \in \mathbf{C}$, $t_{\text{2D}} \in \mathbf{T}_{\text{2D}}$, $t_{\text{3D}} \in \mathbf{T}_{\text{3D}}$, $\theta \in \boldsymbol{\Theta}$ and $\omega \in \boldsymbol{\Omega}$. The regression targets for 2D ground truths (GTs) $[\hat{x}, \hat{y}, \hat{w}, \hat{h}]_{\text{2D}}$ are defined as:

$$\hat{t}_{x\text{2D}} = \frac{\hat{x}_{\text{2D}} - i}{\Phi_{w\text{2D}}},\ \ \hat{t}_{y\text{2D}} = \frac{\hat{y}_{\text{2D}} - j}{\Phi_{h\text{2D}}},\ \ \hat{t}_{w\text{2D}} = \log\frac{\hat{w}_{\text{2D}}}{\Phi_{w\text{2D}}},\ \ \hat{t}_{h\text{2D}} = \log\frac{\hat{h}_{\text{2D}}}{\Phi_{h\text{2D}}}, \tag{5.1}$$

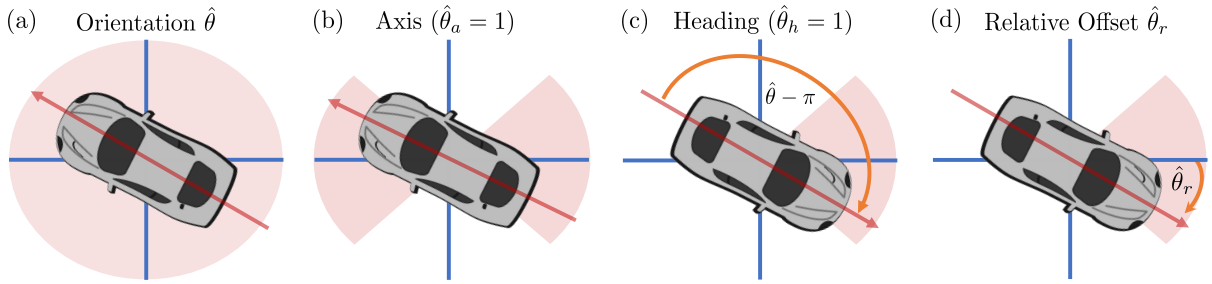

Figure 5.3: **Orientation.** Our proposed orientation formulation decomposes an object orientation $\hat{\theta}$ (a) into an axis classification $\hat{\theta}_a$ (b), a heading classification $\hat{\theta}_h$ (c), and an offset $\hat{\theta}_r$ (d). Our method disentangles the objectives of axis and heading classification while greatly reducing the offset region (red) by a factor of $\frac{1}{4}$.

where $(i, j) \in \mathbb{R}^{w \times h}$ represent the pixel coordinates of the corresponding box. Similarly, following the equation of $\hat{z} \cdot [\hat{u}, \hat{v}, 1]^{\mathrm{T}}_{\mathrm{2D}} = \boldsymbol{\Upsilon} \cdot [x, y, z, 1]^{\mathrm{T}}_{\mathrm{3D}}$, the regression targets for the projected 3D center GTs are defined as:

$$\hat{t}_u = \frac{\hat{u} - i}{\Phi_{w_{\mathrm{2D}}}}, \qquad \hat{t}_v = \frac{\hat{v} - j}{\Phi_{h_{\mathrm{2D}}}}, \qquad \hat{t}_z = \hat{z} - \Phi_z. \tag{5.2}$$

Lastly, the regression targets for 3D dimensions GTs $[\hat{w}, \hat{h}, \hat{l}]_{\mathrm{3D}}$ are defined as:

$$\hat{t}_{w_{\mathrm{3D}}} = \log \frac{\hat{w}_{\mathrm{3D}}}{\Phi_{w_{\mathrm{3D}}}}, \qquad \hat{t}_{h_{\mathrm{3D}}} = \log \frac{\hat{h}_{\mathrm{3D}}}{\Phi_{h_{\mathrm{3D}}}}, \qquad \hat{t}_{l_{\mathrm{3D}}} = \log \frac{\hat{l}_{\mathrm{3D}}}{\Phi_{l_{\mathrm{3D}}}}. \tag{5.3}$$

The remaining targets for our novel orientation estimation t_{θ_r}, $[\theta_a,\ \theta_h]$, and 3D self-balancing confidence ω are defined in subsequent sections.

5.3.1.3 Orientation Estimation:

We propose a novel object orientation formulation, with a decomposition of three components: axis, heading, and offset (Fig. 5.3). Intuitively, the axis estimation θ_a represents the probability an object is oriented towards the vertical axis ($\theta_a = 0$) or the horizontal axis ($\theta_a = 1$), with its label formally defined as: $\hat{\theta}_a = |\sin\hat{\theta}| < |\cos\hat{\theta}|$, where $\hat{\theta}$ is the ground truth object orientation in radians from a bird's eyes view (BEV) with $[-\pi, \pi)$ bounded range. We then compute an orientation $\hat{\theta}_r$ with a restricted range relative to its axis, *e.g.*, $[-\pi,\ 0)$ when $\hat{\theta}_a = 0$, and $[-\frac{\pi}{2},\ \frac{\pi}{2})$ when $\hat{\theta}_a = 1$. We start with $\hat{\theta}_r = \hat{\theta}$ then add or subtract π from $\hat{\theta}_r$ until the desired range is satisfied.

Intuitively, $\hat{\theta}_r$ loses its heading since the true rotation may be $\{\hat{\theta}_r,\ \hat{\theta}_r \pm \pi\}$. We therefore preserve the heading using a separate $\hat{\theta}_h$, which represents the probability of $\hat{\theta}_r$ being rotated by π with its GT target defined as:

$$\hat{\theta}_h = \begin{cases} 0 & \hat{\theta} = \hat{\theta}_r \\ 1 & \text{otherwise.} \end{cases} \tag{5.4}$$

Lastly, we encode the orientation offset transformation which is relative to the corresponding anchor, axis, and restricted orientation $\hat{\theta}_r$ as: $\hat{t}_{\theta_r} = \hat{\theta}_r - \Phi_{\theta_a}$. The reverse decomposition is $\theta = \Phi_{\theta_a} + \lfloor \theta_h \rceil \cdot \pi + t_{\theta_r}$ where $\lfloor \rceil$ denotes round.

In designing our orientation, we first observed that the visual difference between objects at opposite headings of $[\theta,\ \theta \pm \pi]$ is low, especially for far objects. In contrast, classifying the axis of an object is intuitively more clear since the visual features correlate with the aspect ratio. Note that $[\theta_a,\ \theta_h,\ \theta_r]$ disentangle these two objectives. Hence, while the axis is being determined, the heading classifier can focus on subtle clues such as windshields, headlights and shape.

We further note that our 2 binary classifications have the same representational power as 4 bins following [81, 90, 103, 114]. Specifically, bins of $[0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}]$. However, it is more common to use considerably more bins (such as 12 in [81]). An important distinction is that bin-based approaches require the network decide axis and heading simultaneously, whereas our method *disentangles* the orientation into the two distinct and explainable objectives. We provide ablations to compare our decomposition and the bin method using $[2, 4, 10]$ bins in Sec. 5.4.4.

5.3.1.4 Self-Balancing Loss:

The novel 3D localization confidence ω follows a self-balancing formulation closely coupled to the network loss. We first define the 2D and 3D loss terms which comprise the general RPN loss. We unroll and match all n_b box estimations to their respective ground truths. A box is matched as foreground when sufficient ($\geq k$) 2D intersection over union (IoU) is met, otherwise it is considered background ($\hat{c} = 0$) and all loss terms except for classification are ignored. The 2D box loss is thus defined as:

$$L_{\text{2D}} = -\log(\text{IoU}(b_{\text{2D}},\ \hat{b}_{\text{2D}}))[\hat{c} \neq 0] + \text{CE}(c,\ \hat{c}), \tag{5.5}$$

where CE denotes a softmax activation followed by logistic cross-entropy loss over the ground truth class $\hat{c}$, and IoU uses predicted b_{2D} and ground truth $\hat{b}_{\text{2D}}$. Similarly, the 3D localization loss for only foreground ($\hat{c} \neq 0$) is defined as:

$$L_{\text{3D}} = L_1(t_{\text{3D}},\ \hat{t}_{\text{3D}}) + \lambda_a \cdot \text{BCE}([\theta_a,\ \theta_h],\ [\hat{\theta}_a,\ \hat{\theta}_h]), \tag{5.6}$$

where BCE denotes a sigmoid activation followed by binary cross-entropy loss. Next we define the final self-balancing confidence loss with the ω estimation as:

$$L = L_{\text{2D}} + \omega \cdot L_{\text{3D}} + \lambda_L \cdot (1 - \omega), \tag{5.7}$$

where λ_L is the rolling mean of the n_L most recent L_{3D} losses per mini-batch. Since ω is predicted per-box via a sigmoid, the network can intuitively balance whether to use the loss of L_{3D} or incur a proportional penalty of $\lambda_L \cdot (1 - \omega)$. Hence, when the confidence is high ($\omega \approx 1$) we infer that the network is confident in its 3D loss L_{3D}. Conversely, when the confidence is low ($\omega \approx 0$), the network is uncertain in L_{3D}, thus incurring a flat penalty is preferred. At inference, we fuse the self-balancing confidence with the classification score as $\mu = c \cdot \omega$.

The proposed self-balancing loss has two key benefits. Firstly, it produces a useful 3D localization confidence with inherent correlation to 3D IoU (Sec. 5.4.4). Secondly, it enables the network to re-balance samples which are exceedingly challenging and re-focus on the more reasonable targets. Such a characteristic can be seen as the inverse of hard-negative mining, which is important while monocular 3D object detection remains highly difficult and unsaturated (Sec. 5.4.1).

5.3.2 Ego-motion

A challenge with the dynamics of urban scenes is that not only are most foreground objects in motion, but the capturing camera itself is dynamic. Therefore, for a full understanding of the scene dynamics, we design our model to additionally predict the self-movement of the capturing camera, *e.g.*, ego-motion.

We define ego-motion in the conventional six degrees of freedom: translation $[\gamma_x, \gamma_y, \gamma_z]$ in meters and rotation $[\rho_x, \rho_y, \rho_z]$ in radians. We attach an ego-motion features layer E with ReLU to the concatenation of two temporally adjacent feature maps χ which is the final layer in our backbone with size $\mathbb{R}^{n_h \times w \times h}$ architecture, defined as $\chi_{t-1} \mid\mid \chi_t$. We then attach predictions for

translation $[\Gamma_x, \Gamma_y, \Gamma_z]$ and rotation $[P_x, P_y, P_z]$, which are of size $\mathbb{R}^{w \times h}$. Instead of using a global pooling, we predict a spatial confidence map $E_c \in \mathbb{R}^{w \times h}$ based on E. We then apply softmax over the spatial dimension of E_c such that $\sum E_c = 1$. Hence, the pooling of prediction maps $[\Gamma,\ P]$ into $[\gamma,\ \rho]$ is defined as:

$$\gamma = \sum_{(i,j)} \Gamma(i,j) \cdot E_c(i,j), \quad \rho = \sum_{(i,j)} P(i,j) \cdot E_c(i,j), \tag{5.8}$$

where (i, j) is the coordinate in $\mathbb{R}^{w \times h}$. We show an overview of the motion features E, spatial confidence E_c, and outputs $[\Gamma,\ P] \rightarrow [\gamma,\ \rho]$ within Fig. 5.2. We use a L_1 loss against GTs $\hat{\gamma}$ and $\hat{\rho}$ defined as $L_{\text{ego}} = L_1(\gamma,\ \hat{\gamma}) + \lambda_r \cdot L_1(\rho,\ \hat{\rho})$.

5.3.3 Kinematics

In order to leverage temporal motion in video, we elect to integrate our RPN and ego-motion into a novel kinematic model. We adopt a 3D Kalman [164] due to its notable efficiency, effectiveness, and interpretability. We next detail our proposed motion model, the procedure for forecasting, association, and update.

5.3.3.1 Motion Model:

The critical variables we opt to track are defined as 3D center $[\tau_x, \tau_y, \tau_z]$, 3D dimensions $[\tau_w, \tau_h, \tau_l]$, orientation $[\tau_\theta, \tau_{\theta_h}]$, and scalar velocity τ_v. We define τ_θ as an θ orientation constrained to the range of $[-\frac{\pi}{2}, \frac{\pi}{2})$, and θ_h as in Sec. 5.3.1[1]. We constrain the motion model to only allow objects to move in the direction of their orientation. Hence, we define the state transition $\mathbf{F} \in \mathbb{R}^{9 \times 9}$ as:

[1] We do not use the axis θ_a of Sec. 5.3.1, since we expect the orientation to change *smoothly* and do not wish the orientation is *relative* to a (potentially) changing axis.

$$\mathbf{F} = \begin{bmatrix} & \cos(\tau_\theta + \pi\lfloor\tau_{\theta_h}\rceil) \\ & 0 \\ \mathbf{I}^{9\times 8} & -\sin(\tau_\theta + \pi\lfloor\tau_{\theta_h}\rceil) \\ & 0 \\ & \vdots \\ & 1 \end{bmatrix}, \tag{5.9}$$

where $\mathbf{I}$ denotes the identity matrix and the state variable order is $[\tau_x, \tau_y, \tau_z, \tau_w, \tau_h, \tau_l, \tau_\theta, \tau_{\theta_h}, \tau_v]$. We constrain the velocity to only move within its orientation to simplify the Kalman to work more effectively. Recall that since our measurement model RPN processes a single frame, it does *not* measure velocity. Thus, to map the tracked state space and measurement space, we also define an observation model $\mathbf{H}$ as a truncated identity map of size $\mathbf{I} \in \mathbb{R}^{8\times 9}$. We define covariance $\mathbf{P}$ with 3D confidence μ, as $\mathbf{P} = \mathbf{I}^{9\times 9} \cdot (1-\mu) \cdot \lambda_o$ where λ_o is an uncertainty weighting factor. Hence, we avoid the need to manually tune the covariance, while being dynamic to diverse and changing image content.

5.3.3.2 Forecasting:

The forecasting step aims to utilize the tracked state variables and covariances of time $t-1$ to estimate the state of a future time t. The equation to forecast a state variable τ_{t-1} into τ'_t is: $\tau'_t = \mathbf{F}_{t-1} \cdot \tau_{t-1}$, where $\mathbf{F}_{t-1}$ is the state transition model at $t-1$. Note that both objects *and* the capturing camera may have independent motion between consecutive frames. Therefore, we lastly apply the estimated ego-motion to all available tracks' 3D center $[\tau_x, \tau_y, \tau_z]$ by:

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ 1 \end{bmatrix}'_t = \begin{bmatrix} \mathbf{R}, & \mathbf{T} \\ 0, & 1 \end{bmatrix}^t_{t-1} \cdot \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ 1 \end{bmatrix}'_t, \quad \tau'_{t\theta} = \tau'_{t\theta} + \rho_y \tag{5.10}$$

where $\mathbf{R}^t_{t-1} \in \mathbb{R}^{3\times3}$ denotes the estimated rotation matrix converted from Euler angles and $\mathbf{T}^t_{t-1} \in \mathbb{R}^{3\times1}$ the translation vector for ego-motion (as in Sec. 5.3.2). Finally, we forecast a tracked object's covariance $\mathbf{P}$ from $t-1$ to t defined as:

$$\mathbf{P}'_t = \mathbf{F}_{t-1} \cdot \mathbf{P}_{t-1} \cdot \mathbf{F}^T_{t-1} + \mathbf{I}^{9\times9} \cdot (1 - \mu_{t-1}), \tag{5.11}$$

where μ_{t-1} denotes the *average* self-balancing confidence μ of a track's life. Hence, the resultant track states τ'_t and track covariances $\mathbf{P}'_t$ represent the Kalman filter's best forecasted estimation with respect to frame t.

5.3.3.3 Association:

After the tracks have been forecasted from $t-1$ to t, the next step is to associate tracks to corresponding 3D box measurements (Sec. 5.3.1). Let us denote boxes produced by the measurement RPN as $b \in \mathbb{R}^8$ mimicking the tracked state as $[b_x, b_y, b_z, b_w, b_h, b_l, b_\theta, b_{\theta_h}]$[2]. Our matching strategy consists of two steps. We first compute the 3D center distance between the tracks τ'_t and measurements b. The best matches with the lowest distance are iteratively paired and removed until no pairs remain with distance $\leq k_d$. Then we compute the *projected* 2D box IoU between any remaining tracks τ'_t and measurements b. The best matches with the highest IoU are also iter-

[2]We apply the estimated transformations of Sec. 5.3.1 to their respective anchors with equations of [9], and back-project into 3D coordinates to match track variables.

atively paired and removed until no pairs remain with IoU $\geq k_u$. Measured boxes that were *not* matched are added as new tracks. Conversely, tracks that were *not* matched incur a penalty with hyperparameter k_p, defined as $\mu_{t-1} = \mu_{t-1} \cdot k_p$. Lastly, any box who has confidence $\mu_{t-1} \leq k_m$ is removed from the valid tracks.

5.3.3.4 Update:

After making associations between tracks τ'_t and measurements b, the next step is to utilize the track covariance $\mathbf{P}'_t$ and measured confidence μ to update each track to its final state τ_t and covariance $\mathbf{P}_t$. Firstly, we formally define the equation for computing the Kalman gain as:

$$\mathbf{K} = \mathbf{P'}\,\mathbf{H}^T\,(\mathbf{H}\,\mathbf{P'}\,\mathbf{H}^T + \mathbf{I}^{8\times 8}\,(1-\mu)\cdot\lambda_o)^{-1}, \tag{5.12}$$

where $\mathbf{I}^{8\times 8}\,(1-\mu)\cdot\lambda_o$ represents the incoming measurement covariance matrix, and $\mathbf{P'}$ the forecasted covariance of the track. Next, given the Kalman gain $\mathbf{K}$, forecasted state τ'_t, forecasted covariance $\mathbf{P}'_t$, and measured box b, the final track state τ_t and covariance $\mathbf{P}_t$ are defined as:

$$\tau_t = \tau'_t + \mathbf{K}\,(b - \mathbf{H}\,\tau'_t), \qquad \mathbf{P}_t = (\mathbf{I}^{9\times 9} - \mathbf{K}\,\mathbf{H})\,\mathbf{P}'_t. \tag{5.13}$$

We lastly aggregate each track's overall confidence μ_t over time as a running average of $\mu_t = \frac{1}{2}\cdot(\mu_{t-1}+\mu)$, where μ is the measured confidence.

5.3.4 Implementation Details

Our framework is implemented in PyTorch [125], with the 3D RPN settings of [9]. We release source code at `http://cvlab.cse.msu.edu/project-kinematic.html`. We use a batch size of 2 and learning rate of 0.004. We set $k = 0.5$, $\lambda_o = 0.2$, $\lambda_r = 40$, $n_L = 100$, $\lambda_a = k_u =$

0.35, $k_d = 0.5$, $k_p = 0.75$, and $k_m = 0.05$. To ease training, we implement three phases. We first train the 2D-3D RPN with $L = L_{\text{2D}} + L_{\text{3D}}$, then the self-balancing loss of Eq. 5.7, for $80k$ and $50k$ iterations. We freeze the RPN to train ego-motion using L_{ego} for $80k$. Our backbone is DenseNet121 [67] where $n_h = 1{,}024$. Inference uses 4 frames as provided by [51].

5.4 Experiments

We benchmark our kinematic framework on the KITTI [51] dataset. We comprehensively evaluate on 3D Object Detection and Bird's Eye View (BEV) tasks. We then provide ablation experiments to better understand the effects and justification of our core methodology. We show qualitative examples in Fig. 5.6.

5.4.1 KITTI Dataset

The KITTI [51] dataset is a popular benchmark for self-driving tasks. The official dataset consists of $7{,}481$ training and $7{,}518$ testing images including annotations for 2D/3D objects, ego-motion, and 4 temporally adjacent frames. We evaluate on the most widely used validation split as proposed in [24], which consists of $3{,}712$ training and $3{,}769$ validation images. We focus primarily on the car class.

5.4.1.1 Metric:

Average precision (AP) is utilized for object detection in KITTI. Following [140], the KITTI metric has updated to include 40 ($\uparrow 11$) recall points while *skipping* the first. The AP_{40} metric is more stable and fair overall [140]. Due to the official adoption of AP_{40}, it is not possible to compute AP_{11} on test. Hence, we elect to use the AP_{40} metric for all reported experiments.

	AP_{3D} (IoU ≥ 0.7)			AP_{BEV} (IoU ≥ 0.7)			s/im*
	Easy	Mod	Hard	Easy	Mod	Hard	
FQNet [103]	2.77	1.51	1.01	5.40	3.23	2.46	0.50†
ROI-10D [114]	4.32	2.02	1.46	9.78	4.91	3.74	0.20
GS3D [90]	4.47	2.90	2.47	8.41	6.08	4.94	2.00†
MonoPSR [81]	10.76	7.25	5.85	18.33	12.58	9.91	0.20
MonoDIS [140]	10.37	7.94	6.40	17.23	13.19	11.12	–
M3D-RPN [9]	14.76	9.71	7.42	21.02	13.67	10.23	*0.16*
AM3D [113]	*16.50*	*10.74*	**9.52**	*25.03*	*17.32*	**14.91**	0.40
Ours	**19.07**	**12.72**	*9.17*	**26.69**	**17.52**	*13.10*	**0.12**

Table 5.1: **KITTI Test**. We compare with SOTA methods on the KITTI test dataset. We report performances using the AP_{40} [140] metric available on the official leaderboard. * the runtime is reported from the official leaderboard with slight variances in hardware. We indicate methods reported on CPU with †. **Bold**/*italics* indicate best/second AP.

	AP_{3D} (IoU $\geq [0.7/0.5]$)			AP_{BEV} (IoU $\geq [0.7/0.5]$)		
	Easy	Mod	Hard	Easy	Mod	Hard
MonoDIS [140]	11.06/ –	7.60/ –	6.37/ –	18.45/ –	12.58/ –	10.66/ –
M3D-RPN [9]	*14.53/48.56*	*11.07/35.94*	*8.65/28.59*	*20.85/53.35*	*15.62/39.60*	*11.88/31.77*
Ours	**19.76/55.44**	**14.10/39.47**	**10.47/31.26**	**27.83/61.79**	**19.72/44.68**	**15.10/34.56**

Table 5.2: **KITTI Validation**. We compare with SOTA on KITTI validation [24] split. Note that methods published prior to [140] are unable to report the AP_{40} metric.

5.4.2 3D Object Detection

We evaluate our proposed framework on the task of 3D object detection, which requires objects be localized in 3D camera coordinates as well as supplying the 3D dimensions and BEV orientation relative to the XZ plane. Due to the strict requirements of IoU in three dimensions, the task demands precise localization of an object to be considered a match (3D IoU ≥ 0.7). We evaluate our performance on the official test [51] dataset in Tab. 5.1 and the validation [24] split in Tab. 5.2.

We emphasize that our method improves the SOTA on KITTI test by a significant margin of $\uparrow 1.98\%$ compared to [113] on the moderate configuration with IoU ≥ 0.7, which is the most common metric used to compare. Further, we note that [113] require multiple encoder-decoder

networks which add overhead compared to our single network approach. Hence, their runtime is $\approx 3\times$ (Tab. 5.1) compared to ours, self-reported on *similar* but not identical GPU hardware. Moreover, [9] is the most comparable method to ours as both utilize a single network and an RPN archetype. We note that our method significantly outperforms [9] and many other recent works [81, 90, 103, 114, 140] by $\approx 3.01 - 11.21\%$.

We further evaluate our approach on the KITTI validation [24] split using the AP_{40} for available approaches and observe similar overall trends as in Tab. 5.2. For instance, compared to competitive approaches [9, 140] our method improves the performance by $\uparrow 3.03\%$ for the challenging IoU criteria of ≥ 0.7. Similarly, our performance on the more relaxed criteria of IoU ≥ 0.5 increases by $\uparrow 3.53\%$. We additionally visualize detailed performance characteristics on AP_{3D} at discrete depth $[15,\ 30,\ \text{All}]$ meters and IoU matching criterias $0.3 \rightarrow 0.7$ in Fig. 5.4.

5.4.3 Bird's Eye View

The Bird's Eye View (BEV) task is similar to 3D object detection, differing primarily in that the 3D boxes are firstly projected into the XZ plane then 2D object detection is calculated. The projection collapses the Y-axis degree of freedom and intuitively results in a less precise but reasonable localization.

We note that our method achieves SOTA performance on the BEV task regarding the moderate setting of the KITTI test dataset as detailed in Tab. 5.1. Our method performs favorably compared with SOTA works [9, 81, 90, 103, 114, 140] (*e.g.*, $\approx 3.85 - 14.29\%$), and similarly to [113] at a notably lower runtime cost. We suspect that our method, especially the self-balancing confidence (Eq. 5.7), prioritizes precise localization which warrants more benefit in **full** 3D Object Detection task compared to the Bird's Eye View task.

Our method performs similarly on the validation [24] split of KITTI (Tab. 5.2). Specifically,

	AP_{3D} (IoU $\geq [0.7/0.5]$)			AP_{BEV} (IoU $\geq [0.7/0.5]$)		
	Easy	Mod	Hard	Easy	Mod	Hard
Baseline	13.81/47.10	9.71/34.14	7.44/26.90	20.08/52.57	13.98/38.45	11.10/29.88
+ θ decomposition	16.66/51.47	12.10/38.58	9.40/30.98	23.15/56.48	17.43/42.53	13.48/34.37
+ self-confidence	16.64/52.18	12.77/38.99	9.60/**31.42**	24.22/58.52	18.02/42.95	13.92/**34.80**
+ $\mu = c \cdot \omega$	*18.28/54.70*	*13.55/39.33*	*10.13*/31.25	*25.72/60.87*	*18.82/44.36*	*14.48*/34.48
+ kinematics	**19.76/55.44**	**14.10/39.47**	**10.47**/*31.26*	**27.83/61.79**	**19.72/44.68**	**15.10**/*34.56*

Table 5.3: **Ablation Experiments**. We conduct a series of ablation experiments with the validation [24] split of KITTI, using diverse IoU matching criteria of $\geq 0.7/0.5$.

compared to [9, 140] our proposed method outperforms by a range of $\approx 4.10 - 7.14\%$, which is **consistent** to the same methods on test $\approx 3.85 - 4.33\%$.

5.4.4 Ablation Study

To better understand the characteristics of our proposed kinematic framework, we perform a series of ablation experiments and analysis, summarized in Tab. 5.3. We adopt [9] without hill-climbing or depth-aware layers as our baseline method. Unless otherwise specified we use the experimental settings outlined in Sec. 5.3.4.

5.4.4.1 Orientation Improvement:

The orientation of objects is intuitively a critical component when modeling motion. When the orientation is decomposed into axis, heading, and offset the overall performance significantly improves, *e.g.*, by $\uparrow 2.39\%$ in AP_{3D} and $\uparrow 3.45\%$ in AP_{BEV}, as detailed within Tab. 5.3. We compute the mean angle error of our baseline, orientation decomposition, and kinematics method which respectively achieve 13.4°, 10.9°, and 6.1° ($\downarrow 54.48\%$), suggesting our proposed methodology is significantly more *stable*.

We compare our orientation decomposition to bin-based methods following general idea of [81, 90, 103, 114]. We specifically change our orientation definition into $[\theta_b,\ \theta_o]$ which includes a bin

classification and an offset. We experiment with the number of bins set to $[2,\ 4,\ 10]$ which are uniformly spread from $[0,\ 2\pi)$. Note that 4 bins have the same representational power as using binary $[\theta_a,\ \theta_h]$. We observe that the ablated bin-based methods achieve $[9.47\%,\ 10.02\%,\ 10.76\%]$ in AP_{3D}. In comparison, our decomposed orientation achieves 12.10% in AP_{3D}. We provide additional detailed experiments in our supplemental material.

Further, we find that our proposed kinematic motion model (as in Sec. 5.3.3) *degrades* in performance when a comparatively erratic baseline (Row 1. Tab. 5.3) orientation is utilized instead ($14.10 \rightarrow 11.47$ on AP_{3D}), reaffirming the importance of having a consistent/stable orientation when aggregating through time.

5.4.4.2 Self-balancing Confidence:

We observe that the self-balancing confidence is important from two key respects. Firstly, its integration in Eq. 5.7 enables the network to re-weight box samples to focus more on reasonable samples and incur a flat penalty (*e.g.*, $\lambda_L \cdot (1-\omega)$ of Eq. 5.7) on the difficult samples. In a sense, the self-balancing confidence loss is the *inverse* of hard-negative mining, allowing the network to focus on reasonable estimations. Hence, the loss on its own improves performance for AP_{3D} by $\uparrow 0.67\%$ and AP_{BEV} by $\uparrow 0.59\%$.

The second benefit of self-balancing confidence is that by design ω has an inherent correlation with the 3D object detection performance. Recall that we fuse ω with the classification score c to produce a final box rating of $\mu = c \ \cdot \ \omega$, which results in an additional gain of $\uparrow 0.78\%$ in AP_{3D} and $\uparrow 0.80\%$ in AP_{BEV}. We further analyze the correlation of μ with 3D IoU, as is summarized in Fig. 5.5. The correlation coefficient with the classification score c is significantly lower than the correlation using μ instead (0.301 vs. 0.417). In summary, the use of the Eq. 5.7 and μ account for a gain of $\uparrow 1.45\%$ in AP_{3D} and $\uparrow 1.39\%$ in AP_{BEV}.

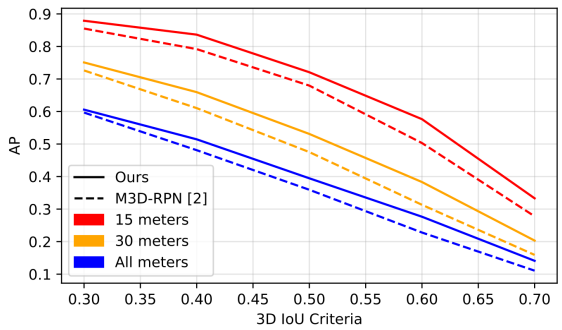

Figure 5.4: We compare AP_{3D} with [9] by varying 3D IoU criteria *and* depth.

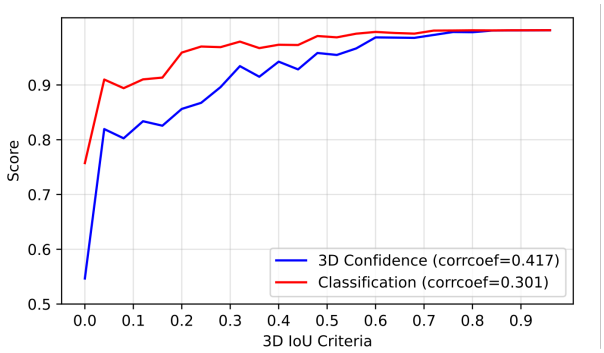

Figure 5.5: We show the correlation of 3D IoU to classification c and 3D confidence μ.

5.4.4.3 Temporal Modeling:

The use of video and kinematics is a significant motivating factor for this work. We find that the use of kinematics (detailed in Sec. 5.3.3) results in a gain of $\uparrow 0.55\%$ in AP_{3D} and $\uparrow 0.90\%$ in AP_{BEV}, as shown in Tab. 5.3. We emphasize that although the improvement is less dramatic versus orientation and self-confidence, the former are important to facilitate temporal modeling. We find that if orientation decomposition and uncertainty are **removed**, by using the baseline orientation and setting μ to be a static constant, then the kinematic performance drastically reduces from $14.10\% \rightarrow 10.64\%$ in AP_{3D}.

We emphasize that kinematic framework not only helps 3D object detection, but also *naturally* produces useful by-products such as velocity and ego-motion. Thus, we evaluate the respective average errors of each motion after applying the camera capture rate to convert the motion into miles per hour (MPH). We find that the per-object velocity and ego-motion speed errors perform reasonably at 7.036 MPH and 6.482 MPH respectively. We depict visual examples of all dynamic by-products in Fig. 5.6 and additionally in supplemental video.

Figure 5.6: **Qualitative Examples.** We depict the image view (left) and BEV (right). We show velocity vector in green, speed and ego-motion in miles per hour (MPH) on top of detection boxes and at the top-left corner, and tracks as dots in BEV.

5.5 Summary

We present a novel kinematic 3D object detection framework which is able to efficiently leverage temporal cues and constraints to improve 3D object detection. Our method naturally provides useful by-products regarding scene dynamics, *e.g.*, reasonably accurate ego-motion and per-object velocity. We further propose novel designs of orientation estimation and a self-balancing 3D confidence loss in order to enable the proposed kinematic model to work effectively. We emphasize that our framework efficiently uses only a *single network* to comprehensively understand a highly

	AP_{3D} (IoU $\geq [0.7/0.5]$)			AP_{BEV} (IoU $\geq [0.7/0.5]$)		
	Easy	Mod	Hard	Easy	Mod	Hard
2 bins	12.83/46.46	9.47/33.78	7.93/26.85	19.17/52.06	14.72/37.54	11.38/31.16
4 bins	12.65/44.01	10.02/33.27	7.87/26.27	19.09/49.86	14.55/37.90	11.14/30.44
10 bins	14.27/49.71	10.74/36.12	8.29/28.62	21.12/54.70	15.37/39.72	11.60/31.75
Our decomp.	16.66/51.47	12.10/38.58	9.40/30.98	23.15/56.48	17.43/42.53	13.48/34.37

Table 5.4: **Orientation**. We compare our orientation decomposition to bin-based orientation following the high-level concepts within [81, 90, 103, 114], using AP_{3D} and AP_{BEV}. We evaluate our performances on the KITTI validation set [24] using IoU $\geq 0.7/0.5$.

dynamic 3D scene for urban autonomous driving. Moreover, we demonstrate our method's effectiveness through detailed experiments on the KITTI [51] dataset across the 3D object detection and BEV tasks.

Acknowledgments: Research was partially sponsored by the Army Research Office under Grant Number W911NF-18-1-0330. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. This work is further partly funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 409792180 (Emmy Noether Programme, project: Real Virtual Humans).

5.6 Supplementary Material

5.6.1 Orientation Ablations

We provide detailed experiments on 3D object detection and Bird's Eye View tasks to compare our orientation decomposition performance with bin-based approaches such as [81, 90, 103, 114]

within Tab. 5.4. Recall that bin-based orientation first classifies the best bin for orientation then predicts an offset with respect to the bin. In contrast, our method disentangles the bin classification into a distinct explainable objectives such as an axis classification and a heading classification. For such experiments we change our formulation to use bins of $[2,\ 4,\ 10]$, where 4 bins has a similar representational power as two binary classifications $[\theta_a,\ \theta_h]$. The bins are spread uniformly from $[0,\ 2\pi]$ and an offset is predicted afterwards. We use the settings in Sec. 3.4 in main paper. We emphasize that our method outperforms the bin-based approaches between $\approx 1.36 - 2.63\%$ on AP_{3D} and $\approx 2.06 - 2.71\%$ on AP_{BEV} using the standard moderate setting and IoU ≥ 0.7.

5.6.2 Kalman Forecasting

Since our method uses ego-motion and a 3D Kalman filter to aggregate temporal information, the approach can be modified to act as a box forecaster. Although our method was not strictly designed for the tracking and forecasting task, we evaluate the 3D object detection and Bird's Eye View performance after forecasting $n_f = [1,\ 2,\ 3,\ 4]$ frames into the future. We assume a static ego-motion for unknown frames and otherwise use the Kalman equations described in the main paper Sec. 3.3 to forecast the tracked boxes.

For all forecasting experiments we process 4 temporally adjacent frames before forecasting. Since KITTI only provides a current frame and 3 proceeding frames, we carefully map images back to the raw dataset in order to forecast. For instance, when $n_f = 2$ we infer using frames $[-5, -4, -3, -2]$ then forecast ego-motion and Kalman n_f times. We then evaluate with respect to frame 0 which is the standard timestamp KITTI provides images and 3D labels for. We provide detailed performances on AP_{3D} in Tab. 5.5 and AP_{BEV} in Tab. 5.6. We find that the forecasting performance degrades through time but performs reasonably $1-2$ frames ahead, being competitive in magnitude to state-of-the-art methods on the KITTI test dataset as reported in Tab. 1 of the main

	AP_{3D} (IoU $\geq [0.7/0.5/0.3]$)		
	Easy	Mod	Hard
Forecast $\rightarrow 4$	1.16 / 18.47 / 47.26	0.84 / 11.21 / 29.22	0.62 / 8.97 / 23.40
Forecast $\rightarrow 3$	3.72 / 28.97 / 58.46	2.32 / 18.05 / 37.82	1.75 / 13.88 / 29.80
Forecast $\rightarrow 2$	7.84 / 39.40 / 68.87	5.10 / 25.48 / 48.30	4.14 / 20.20 / 37.84
Forecast $\rightarrow 1$	16.09 / 49.66 / 75.88	10.64 / 34.18 / 55.26	8.14 / 26.62 / 44.01
No Forecast	19.76 / 55.44 / 79.81	14.10 / 39.47 / 60.57	10.47 / 31.26 / 48.95

Table 5.5: **Forecasting - 3D Object Detection**. We evaluate our forecasting performance on AP_{3D} within the KITTI validation [24] set and using IoU $\geq 0.7/0.5/0.3$.

	AP_{BEV} (IoU $\geq [0.7/0.5/0.3]$)		
	Easy	Mod	Hard
Forecast $\rightarrow 4$	5.48 / 29.40 / 54.52	3.54 / 18.13 / 36.13	2.90 / 14.71 / 28.49
Forecast $\rightarrow 3$	11.03 / 39.08 / 64.87	6.89 / 24.01 / 43.52	5.67 / 18.85 / 34.91
Forecast $\rightarrow 2$	17.02 / 47.07 / 72.33	10.76 / 31.62 / 51.67	8.37 / 25.47 / 40.79
Forecast $\rightarrow 1$	23.58 / 55.99 / 77.48	15.79 / 39.33 / 58.05	12.54 / 31.22 / 46.59
No Forecast	27.83 / 61.79 / 81.20	19.72 / 44.68 / 63.44	15.10 / 34.56 / 49.84

Table 5.6: **Forecasting - Bird's Eye View**. We evaluate our forecasting performance on AP_{BEV} within the KITTI validation [24] set and using IoU $\geq 0.7/0.5/0.3$.

paper. For instance, forecasting 1 and 2 frames results in 10.64% and 5.10% AP_{3D} respectively, which are competitive to methods [9, 81, 90, 103, 113, 114, 140] on the test dataset.

5.6.3 Qualitative Video

We further provide a qualitative demonstration video at `http://cvlab.cse.msu.edu/project-kinematic.html`. The video demonstrates our framework's ability to determine a full scene understanding including 3D object cuboids, per-object velocity and ego-motion. We compare to a related monocular work of M3D-RPN [9], plot ground truths, image view, Bird's Eye View, and the track history.

Chapter 6

3D Mesh Discovery in the Wild

Predicting 3D object shapes from a single image is a seminal task in computer vision with applications in embodied AI and content creation. Prior work either experiment on synthetic benchmarks, where 3D annotations are available, or avoid 3D supervision by carefully tuning for specific object categories with the help of 2D pose annotations and expert-designed 3D priors. While groundbreaking, these methods cannot scale to many object categories and complex datasets. In this work, we propose a general end-to-end approach to discover 3D shapes from real-world scenes. Our approach, 3DMD, is trained on a collection of unassociated images using only 2D masks as supervision. At test time, 3DMD infers the 3D object shape from a single image in view coordinates. By avoiding object-specific priors and external 3D knowledge, 3DMD scales to $10\times$ more categories than prior work. We test our approach on challenging real-world datasets, such as Pascal3D+, CUB-200 where we outperform methods which use more supervision or hand-crafted priors. Most importantly, we show results on COCO where we scale to dozens of object categories.

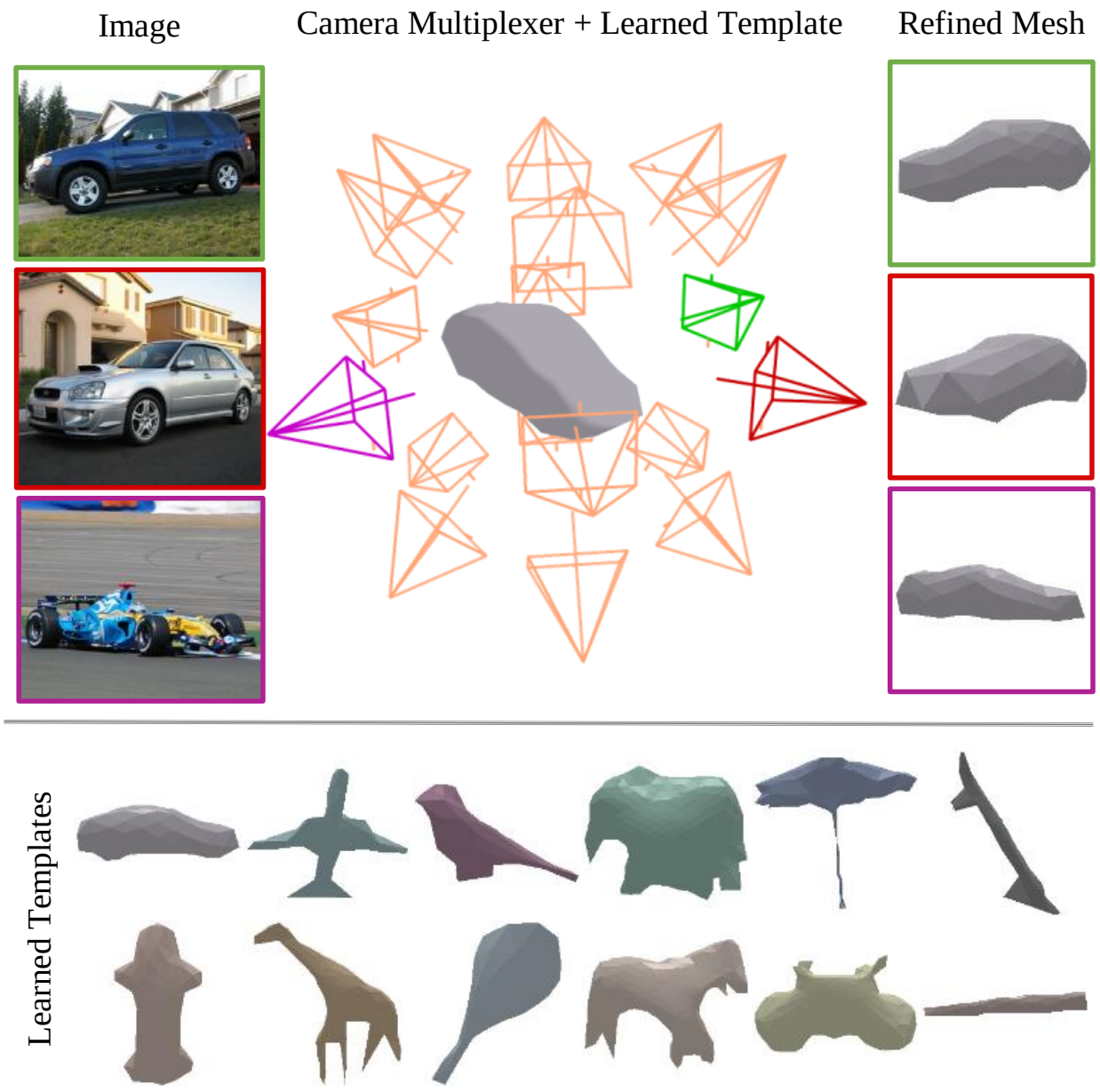

Figure 6.1: Our approach 3D Mesh Discovery (3DMD) takes a collection of images as input and jointly learns a shared 3D mesh template and refined meshes over a diverse set of object categories.

6.1 Introduction

Inferring 3D object shapes from monocular inputs is a seminal goal of visual intelligence, with real-world applications in robotics [86, 136, 176], content creation [7, 8, 34, 165], and self-driving

cars [81, 113]. Equivalent 2D tasks such as instance [63, 64, 77] and semantic segmentation [111] have seen rapid progress due to the success of supervised learning [65, 79, 141] and the advent of large-scale annotated datasets [101]. In the case of 3D reconstruction, supervised methods [56, 87, 168] have shown promising results but cannot scale to hundreds of object categories due to the challenges in collecting 3D shape annotations at scale.

Without 3D supervision, 2D cues must be exploited to predict 3D shapes. Reasoning about 3D from 2D is well studied in computer vision, commonly with multi-view inputs or in predicting depth from motion [1, 33, 57, 182] or disparity [4, 145, 153, 163]. Some pioneering recent efforts [58, 72, 83, 121] have predicted 3D shapes from monocular inputs, training on image collections. Without 3D supervision or multi-view inputs, these methods rely heavily on object category templates, 2D keypoints and hand-crafted object priors. This external category-driven knowledge is another form of supervision which limits scalability.

In this work we propose an end-to-end framework for 3D mesh discovery from a single image, which we call 3DMD. Our approach does not require 3D supervision, multi-view inputs, hand-designed category templates or 2D pose annotations and thus scales to many object categories and in the wild settings. We build on two critical observations:

First, instances of the same object category have similar shapes. We relate object instances by associating them to a learned canonical template shape. Template learning is aided by selecting an image to serve as the *reference sample* for the category. We predict per-instance meshes by posing and deforming the learned template. While prior work uses expert-designed templates [58, 72], our template learning enables better scalability to diverse categories.

Second, an object's posed 3D shape and 2D appearance are consistent. Hence, our predicted 3D shape is taught to agree with the 2D appearance with the help of differentiable mesh rendering [22, 35, 74, 105, 112, 129]. 2D silhouettes are commonly used to encourage such agreement [58, 72, 83].

However, 2D silhouettes alone are not discriminative enough to ground the object pose, *e.g*. a front and a back facing car have similar 2D masks. Therefore, we partition the 3D shape into clusters then force a greater consistency in a self-supervised manner.

Our approach, 3DMD, consists of three components: a template learner, a camera multiplexer [58, 83] and a mesh refinement branch. The template learner estimates a shared 3D shape for a category from a collection of images, using the *reference sample* for a canonical representation. The camera multiplexer predicts the pose of the object in the input image, where the identity pose is defined by the *reference sample*. Unlike [58], our camera multiplexer spreads over the unit sphere and remains active during training and inference. Lastly, the mesh refinement branch deforms the transformed 3D shape to match the appearance of the object in the input image, addressing the intra-class variability. The result is a 3D object mesh in *view coordinates* predicted from a monocular image. An overview of our multiplexer, refined meshes, and learned templates is shown in Fig. 6.1.

We test our method on three real-world datasets: Pascal3D+ [170], CUB-200 [158] and COCO [101]. We benchmark our approach on Pascal3D+, which contains ground truth mesh models, and demonstrate competitive 3D reconstruction from previous state of the art despite using less 3D priors or supervision. In addition, incorporating priors used by prior work into our method further improves performance. We expand our results to tens of categories, an important advancement from prior works. We show exciting quantitative and qualitative results for challenging in the wild objects within COCO [101] such as umbrellas, giraffes, elephants, skateboards and many more.

6.2 Related Work

Our method predicts a 3D mesh in view coordinates from a single input image, only using 2D masks for supervision. We provide background on techniques for image-based supervised and weakly supervised 3D shape reconstruction.

Supervised 3D Reconstruction Supervised methods can utilize a variety of 3D shape representations including: implicit functions [22,102,116,124,142], which require post-processing to extract a 3D triangle mesh, 3D point clouds [45,97], voxels occupancy grids [29,167,168], and 3D meshes deformed from a template sphere [133,143,144,159]. Gkioxari et al. [56] utilize a combination of voxelization and 3D mesh deformation. The majority of these methods show results on synthetic datasets, such as ShapeNet [19], or small scale real-world benchmarks, such as Pix3D [149], for which 3D supervision is available in the form of CAD models. In this work, we focus on real-world large-scale image datasets, such as COCO [101] and Pascal3D+ [170].

Our approach uses 3D triangle meshes deformed relative to *learned* 3D template shapes. We find the triangle mesh structure to be efficient and effective at modeling shared category-specific features via a global template, and instance-level features using graph convolution [56, 159].

Weakly Supervised 3D Reconstruction Understanding 3D objects using only images and weak 2D supervision [58, 72, 73, 92, 156] is a highly ill-posed problem. 2D supervision commonly takes the form of an object segmentation mask or 2D keypoints. However, ambiguities arise when learning 3D shape from 2D supervision alone. To overcome these ambiguities, methods introduce other forms of supervision and priors to constrain the solution space.

Some techniques [129, 155, 156, 173] require multiple views of the same instance of an object. This limits the scope to pairs of images and prevents in the wild scenarios. Others [72, 73] use 2D keypoints to predict the pose. However, keypoints are non-trivial to annotate, and are not

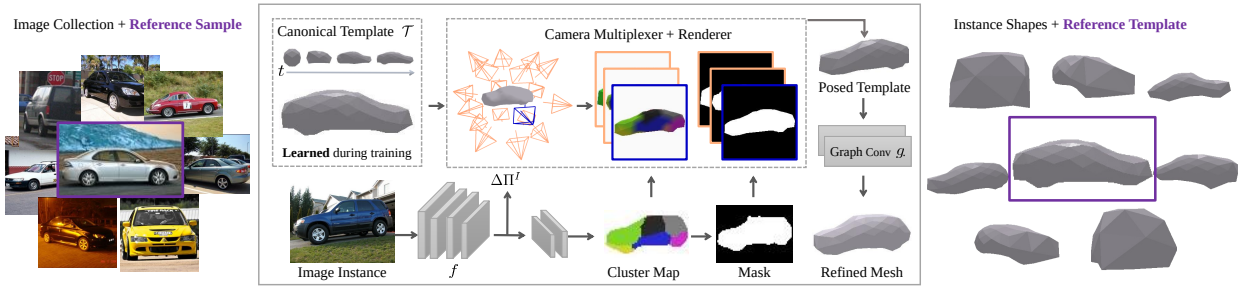

Figure 6.2: Framework overview of 3DMD. Our method learns a *shared* 3D mesh template $\mathcal{T}$ starting with a unit sphere and canonical to a reference sample. Then a camera multiplexer determines the best pose relative to network predictions of object partitions and mask. Finally, the best pose is used to refine the template mesh to match the image inputs (left) producing instance-level mesh shapes (right).

applicable to a large number of object categories. Goel et al. [58] eliminate the need for keypoints by using a camera-multiplex to make pose estimations over a constrained prior distribution. Our approach also leverages a multiplexer. In contrast to [58], ours is free of hand-crafted priors and uses the same camera distribution for all categories or datasets. Moreover, ours functions in an online fashion rather than being memorized independently per training sample in [58].

Moreover, [58, 72, 82] leverage expert 3D template meshes to initialize the shape prediction. The templates are most commonly designed by humans, and capture critical priors such as object structure, relative dimensions, as well as giving *meaning* to the canonical pose. Although powerful, we consider external templates to be strong 3D supervision which in turn limits scalability to many categories.

Recent work by Li et al. [92] utilize a part segmentation network [68] to help constrain pose. In contrast to [92], we propose an *end-to-end* framework which does not require pre-training a separate part segmentation network. Our method uses a single model to jointly learn a shared 3D template, instance specific object pose, and refined shape. Concurrent work [178] use a voxel and mesh approach built on HoloGAN [120] to reconstruct objects with only masks.

We demonstrate our generalization on many categories from Pascal3D+ [170], CUB-200 [158],

and COCO [101]. Compared to prior works [58, 72, 73, 92, 156], we evaluate thoroughly on all 12 Pascal3D+ in the wild categories.

6.3 Method

Our goal is to reconstruct 3D objects from a collection of 2D images *without* 3D supervision. For each object category, we jointly learn a shared template shape and instance-level mesh refinement to match the unique appearance of the input image. At train time, *only* 2D instance-level masks are used. At test time, our model takes a single RGB image as input and reconstructs the object's 3D shape in *view coordinates*. Our approach, 3DMD, scales to many object categories in real-world complex scenes with diverse lighting and pose. Fig. 6.2 shows an overview of our framework.

We first detail our template learner powered by a uniform camera multiplexer, which comprise two enabling components of our approach. Our camera multiplexer covers the full range of poses, is free of manually set object priors and is driven by a self-supervised cluster consistency loss to regularize the object pose. We lastly explain our instance mesh refinement branch which refines the posed template shape.

6.3.1 Template Learner

An important component of our approach is learning a shared 3D template shape, whose structure closely represents the training object shapes. The template learning process is end-to-end, and leverages differentiable rendering and a uniform camera multiplexer. The 3D template is optimized to fit the full dataset and thus is expected to resemble a category-specific mean shape, in contrast to [58, 72, 82] which rely on expert-designed 3D models. Hence, the template learner enables scaling to many more categories, without the need for expensive 3D prior knowledge.

Template Definition Let us define a deformable template mesh as $\mathcal{T} = (\mathcal{V},\ \mathcal{F})$, where $\mathcal{V}$ denotes a set of learnable vertices and $\mathcal{F}$ a set of faces, fixed to follow a sphere topology. We initialize $\mathcal{T}$ as a unit sphere and constrain vertex movement via an adaptive hyperparameter λ_r.

Camera Multiplexer We aim to learn a 3D template $\mathcal{T}$ to fit *all* training instances of a category. Given an image, we want to estimate a pose for the template that best explains the image. Rather than predicting pose directly, we follow [58, 83] and use a *camera multiplexer* which renders the template from a set of hypothetical poses and selects the pose that minimizes 2D re-projection error.

We use a set of N *shared cameras* $\Pi = \{\pi_1, \ldots, \pi_N\}$ which are distributed uniformly around the template mesh. We constrain the poses in order to render templates tight to the image border, mimicking RoIs in object detection.

Starting from the above camera initializations, we use an image predictor f to estimate instance-level transformations for each camera, denoted as $\Delta\Pi^I = \{\Delta\pi_1^I, \ldots, \Delta\pi_N^I\}$. Hence, this gives $f(I)$ an opportunity to update each camera to a better pose according to the unique image I appearance.

The purpose of our camera multiplexer is to select among many different hypothetical poses and find the *best* pose. We approach camera selection by minimizing two losses: (i) The negative 2D IoU between the i-th rendered pose silhouette S_i and the mask label $\hat{S}$ as $L_{\texttt{IoU}} = -\texttt{IoU}(S_i,\ \hat{S})$. At inference, $\hat{S}$ is produced from a mask prediction branch. We use a differentiable silhouette renderer $\mathcal{R}_S$ to compute per-camera silhouettes following $S_i = \mathcal{R}_S(\mathcal{T},\ \pi_i + \Delta\pi_i^I)$. (ii) A self-supervised cluster consistency loss $L_{\texttt{KL}}$, which measures agreement between pose and image features. Details are described in Sec. 6.3.2. The best pose i^* is selected as the argmin among

camera alignment scores a as:

$$a = (1 + L_{\texttt{IoU}} + \sqrt{L_{\texttt{KL}}}) \text{ and } i^* = \text{argmin}_i \, a_i \tag{6.1}$$

The output of the camera multiplexer is our template $\mathcal{T}$ rotated by the i^*-th camera which best aligns to the object appearance. We denote the best posed template mesh as $\mathcal{P}$. Intuitively, $\mathcal{P}$ best explains 2D shape and cluster partitions.

Our camera multiplexer differs from that of [58] in two key ways. First, [58] uses hand-designed category-specific priors to arrange cameras; in contrast we spread cameras uniformly around the template, aiding scalability to many categories. Second, [58] learns a set of camera poses independently for each training image; this forces them to train a separate pose regressor for use during inference. We instead learn to regress multiplexer poses from a set of shared cameras, enabling end-to-end learning and allowing re-use of the multiplexer during *inference* as well as training.

Template Optimization We have defined our learnable template and our uniform camera multiplexer. Next we discuss the optimization with respect to the template vertices and the network predicted camera transformations $\Delta\Pi^I$.

The vertex positions of the mesh template $\mathcal{T}$ are trained to minimize $L_{\texttt{IoU}}(S_{i^*}, \, \hat{S})$, where i^* denotes the best pose. We optimize the camera transformations $\Delta\Pi^I$ produced from the image predictor f as the equation $\sum_i L_{\texttt{IoU}}(S_i, \, \hat{S})$. Intuitively, the network f is responsible for learning adjustments to the multiplexer poses while the template vertices control the category-specific shared mesh shape. We further regularize the template $\mathcal{T}$ using edge statistics, mesh normal consistency, and laplacian regularization. See supplemental for additional loss optimization details.

Reference Sample Discovering 3D object shapes from images with only 2D supervision is chal-

lenging since infinite 3D shapes can explain a 2D mask. We build on the key insight that object instances from the same object type share similarities in shape. We select a reference image sample from the training set which serves as the canonical interpretation of the object shape. Instances of the same object category are inferred as deformations from the reference shape.

To force our model to ground predictions to the reference sample, we include it in every batch. We optimize $\Delta\pi_1^I$ to be an *identity* transform for the reference sample, and force the multiplexer to output $i^* = 1$ for the reference. Hence, π_1 is tied to the reference's canonical viewpoint. Lastly, we use the reference sample to filter outlier samples which deviate too far from the reference loss magnitude. Fig. 6.6 shows examples of the reference sample and corresponding learned templates across diverse categories.

6.3.2 Cluster Consistency

Jointly learning 3D template shapes and camera poses is challenging when *only* 2D masks serve as supervision. This is because 2D masks are not discriminative enough to signal a uniquely correct 3D shape. Prior works alleviate the ambiguity by relying on human-designed CAD models [58, 72, 82] or by predicting 2D semantic parts [72, 92]. In contrast, we propose a self-supervised cluster consistency loss based only on the posed template $\mathcal{P}$ structure.

Although silhouettes are often not discriminative enough, the image of an object reveals a lot about its pose, for example a right vs left facing car. We encourage our camera-multiplexer to agree with the image content by clustering the learned shared template $\mathcal{T}$ into K clusters, then comparing with a cluster map from the image predictor f, encouraging poses and network features to be in *agreement*.

Defining Clusters We automatically generate clusters on the template vertices $\mathcal{V}$ using K-means

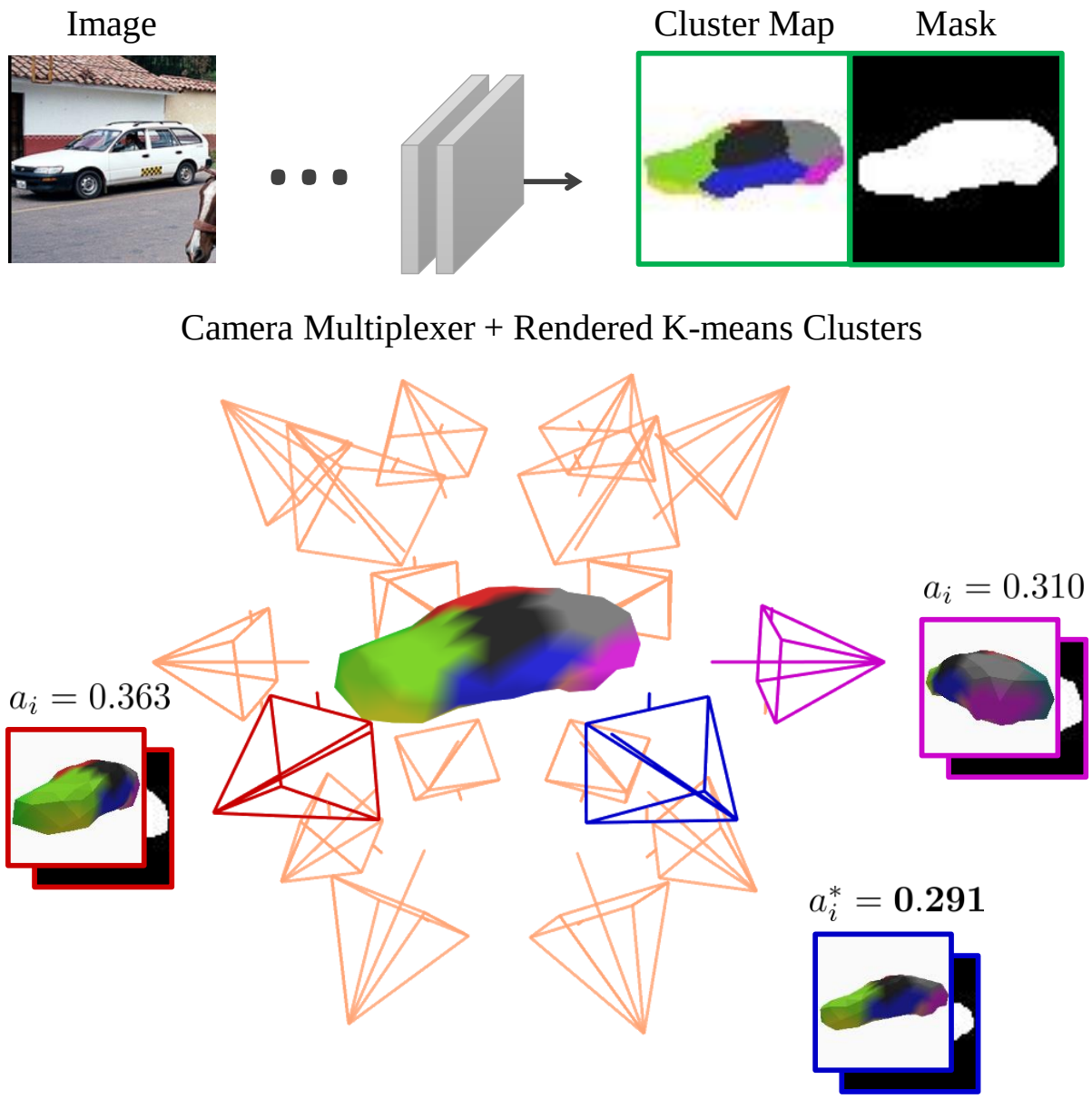

Figure 6.3: Our camera multiplexer computes an alignment score a (Eq. 6.1) which compares network outputs (top) to renderings from uniform cameras (bottom). The cluster maps are compared with rendered clusters using L_{KL} while the 2D mask is used for IoU_{2D}. Multiplexer output is computed as the argmin camera i^* (**bolded**).

[110] with graph distance as the cost. The number of clusters K is set a priori and clustering is updated as $\mathcal{T}$ changes. Note that our clusters are consistent in training but are not directly semantically interpretable. Fig. 6.3 shows example learned clusters.

Cluster Assignments Rather than assigning a vertex to its nearest cluster, we opt for a soft assignment ρ of vertices to a distribution over K clusters. Each vertex is assigned a probability vector which is equal to the softmax of its distance to all cluster centers. We use a differentiable cluster renderer $\mathcal{R}_C$ to compute per-camera clusters defined as $C_i = \mathcal{R}_C(\mathcal{T},\ \rho,\ \pi_i + \Delta\pi_i^I)$. Hence, C denotes the rendered cluster map representing the pixel-wise probability across K clusters and computed for each camera pose.

KL Divergence We have defined how we derive the rendered cluster probabilities C from the camera multiplexer. Next we use our image predictor f to estimate cluster probabilities for each pixel resulting in a softmax cluster map C^I. We compare these cluster maps with the rendered clusters inside the multiplexer using a bi-directional KL-divergence $L_{\mathtt{KL}}$ to encourage the distributions to be consistent.

For the loss to reach zero, the image-based predictions C^I would need to be consistent with the best camera i^*. Similarly, the camera transformations $\Delta\Pi$ would need to morph such that C_{i^*} matches C^I. Fig. 6.3 visualizes both the predicted cluster maps and rendered camera clusters.

6.3.3 Mesh Refinement

Instances of the same object category typically have varying types of shapes, for example a truck vs a hatchback. In addition to learning a shared template and a pose, we further learn to refine an object's 3D shape to correspond to its appearance in the instance-level input image. Note that our mesh refinement is trained end-to-end *with* the template learner, camera multiplexer, and cluster consistency loss.

Following [56, 159], we use graph convolutions to estimate offsets relative to the output of the camera multiplexer. The posed template $\mathcal{P}$ relative to the best camera i^* is used to sample

features from our image predictor f via a vertex alignment sampling. We then define a mesh refiner g comprised of two graph convolutions, following the architecture of [56]. The output of g are offsets for each vertex in $\mathcal{P}$, with a maximum offset constrained to an adaptive hyperparameter λ_o. This results in a uniquely posed *and* deformed 3D mesh for each instance of an object, denoted as $\mathcal{M}$.

At train time, the final refined meshes are rendered using $\mathcal{R}_S$ and optimized against the 2D mask labels, using both the IoU loss and mesh regularizers defined in Sec. 6.3.1.

6.3.4 Adaptive Learning

Our model is tasked to simultaneously learn a shared template shape, camera pose adjustments and instance-level shape without any 3D supervision. These are complex tasks making optimization challenging. To facilitate learning we follow a few simple adaptive learning techniques.

To capture the complexity of varying object categories, our template $\mathcal{T}$ needs sufficient resolution. However, training with too many vertices from random initialization tends to lead to difficulties in optimization. Instead, we start with a small mesh resolution of $|\mathcal{V}| = 42$ then gradually subdivide to $|\mathcal{V}| = 162$, and $|\mathcal{V}| = 642$. We note that we only subdivide if the average template IoU fit is less than 0.80, implying the current level of resolution may be insufficient. However, our mesh refiner g always uses a resolution of 642 vertices even when the template is more coarse.

We *gradually* enable various features of our framework during training. We warmup by optimizing only our template learner and camera multiplexer for 2k iterations. Then we enable the cluster consistency loss. This allows our framework to find a better optimum for the templates and cameras, before the self-supervised consistency loss can serve as a pose regularizer. To prevent a single camera from dominating, we gradually enable the L_{KL} term of Eq. 6.1. After an additional 2k iterations, we enable the mesh refiner, allowing a reasonable template $\mathcal{T}$ to develop first. Finally,

we set the constraint on the template vertex movement λ_r and the mesh refiner vertex movement λ_o adaptively based on 2D mask heuristics. See Suppl. for details.

6.4 Experiments

We tackle the problem of 3D mesh reconstruction from images on three challenging real-world datasets, Pascal3D+ [170], CUB-200 [158] and COCO [101]. We show results on a variety of object categories, such as *giraffes*, *umbrella*, *bicycle* and many more. Note that our approach, 3DMD, scales to many object types and datasets since it only assumes access to 2D silhouettes for supervision and does not require any expert 3D templates, 2D keypoints or manually defined object-specific priors.

Training details We use the PyTorch3D [129] mesh renderer with a resolution of 64×64 and 18 faces per pixel. We train using a batch size of 8. Our initial learning rate is set to 0.04 which we decay by 0.1 at 16k & 24k iterations. We train one model per category for 32k iterations via a SGD optimizer. We set the number of clusters to $K = 12$. We use a total of $N = 48$ cameras. Our input images are of size 128×128, and are filtered to contain unoccluded objects with $< 5\%$ of their border shared with other objects or the image edge. We randomly flip training images. We use the same hyperparameters for *all* categories and datasets.

6.4.1 Pascal3D+

We use the Pascal3D+ [170] dataset containing in the wild images to quantitatively evaluate the 3D mesh reconstruction performance of our method. The dataset provides high-resolution canonical CAD model and pose for $\approx 3{,}000$ objects per category for 12 diverse objects. We summarize our overall performance on each object category in Tab. 6.2.

	Expert template	2D keypoints	Multiple viewpoints	Camera priors	Canonical IoU_{3D} 32	Canonical IoU_{3D} 64	Image View IoU_{3D} 32	Image View IoU_{3D} 64
CSDM [73]		✓			0.600	–	–	–
CMR [72]	✓	✓			0.640	–	–	–
DRC [156]			✓		0.670	–	–	–
U-CMR† [58]	✓			✓	0.680	0.657	0.295	0.265
Li et al. [92]					0.620	–	–	–
3DMD (Ours)					0.658	0.622	0.481	0.450
3DMD+	✓				0.705	0.671	0.443	0.414
3DMD++	✓			✓	**0.718**	**0.678**	**0.522**	**0.493**

Table 6.1: Shape reconstruction performance on Pacal3D+ *car*. We compare our 3DMD with CSDM [73] a keypoint-based approach, CMR [72] which uses expert templates & 2D keypoints, DRC [156] a multi-view volumetric approach, U-CMR [58] which relies on expert templates and dataset-specific camera priors, and Li et al. [92] which uses off-the-shelf part segmentation. Unlike prior works, we report two voxel sizes for a pose-agnostic metric IoU_{3D}^{Can}, *and* a new metric IoU_{3D}^{Im} which factors in pose & shape. † indicates public code results.

Category	IoU_{3D}^{Can} 32	IoU_{3D}^{Can} 64	IoU_{3D}^{Im} 32	IoU_{3D}^{Im} 64
Table	0.200	0.122	0.108	0.070
Bicycle	0.208	0.129	0.166	0.098
Chair	0.248	0.216	0.094	0.069
Aeroplane	0.318	0.263	0.243	0.206
Train	0.392	0.373	0.197	0.174
Boat	0.406	0.362	0.124	0.106
Motorbike	0.453	0.400	0.320	0.283
TV	0.455	0.438	0.390	0.387
Sofa	0.532	0.512	0.403	0.389
Bus	0.533	0.513	0.382	0.358
Car	0.658	0.622	0.481	0.450
Bottle	0.673	0.656	0.520	0.502

Table 6.2: Shape reconstruction results for *all* Pascal3D+ categories.

Metrics To evaluate mesh reconstruction performance, prior works [58, 72, 73, 92, 156] propose a canonical 3D Intersection over Union (IoU_{3D}^{Can}) metric, which measures the voxelized 3D overlap of the *canonical* ground truth and predicted shape. Yet, this metric does not account for the pose. To evaluate for *both* shape and pose, we propose an image-view 3D Intersection over Union (IoU_{3D}^{Im}) metric, which measures the voxelized 3D overlap of the ground truth and predicted shape in *view coordinates*, factoring in pose and shape accuracy. Following prior work, both metrics

use a voxelized approximation for computing 3D IoU and briefly optimize a global scaling and translation factor to account for the general ambiguity of mesh scales.

Comparison with state of the art We compare against a variety of competing methods, which predict 3D shapes without 3D supervision. CSDM [73] uses 2D silhouettes and keypoints to learn deformable basis shapes. DRC [156] is a volumetric approach which relies on multiple viewpoints 2D silhouettes. CMR [72] predicts object meshes from image inputs but requires expert object-specific mesh templates and 2D keypoints. U-CMR [58] also relies on expert object-specific templates but eliminates the need of 2D keypoints by using hand-coded camera hypothesis for each object category, which we refer to as camera priors. These priors work well for a specific object category, *e.g*. images of cars are expected to be taken from a constant height, but fail to generalize to other categories, *e.g*. pictures of airplanes may be captured from cameras below or above. Li et al. [92] use an off-the-shelf external part discovery method and learn a UV map to predict parts of the predicted mesh. The latter is the most comparable to ours, as both learn without 2D keypoints and do not require access to expert object-specific templates. Though, unlike Li et al. [92], our approach is end-to-end and does not depend on an external part segmentation method.

Table 6.1 shows results for *car* in Pascal3D+, the category evaluated unanimously in prior work. Table 6.2 demonstrates comprehensive evaluation for our approach on all categories. We report $\text{IoU}^{\text{Can}}_{\text{3D}}$ & $\text{IoU}^{\text{Im}}_{\text{3D}}$ under two voxel resolutions of 32^3 & 64^3. We draw the following observations. Despite not using camera priors or an expert template, our approach outperforms U-CMR for $\text{IoU}^{\text{Im}}_{\text{3D}}$ (0.481 vs 0.295) and performs on par under the pose-agnostic $\text{IoU}^{\text{Can}}_{\text{3D}}$ (0.658 vs 0.680). Our method outperforms Li et al. [92] (0.658 vs 0.620 $\text{IoU}^{\text{Can}}_{\text{3D}}$), which is directly comparable to ours since they do not use extra 3D priors. Our approach remains consistently superior for both voxelization thresholds. As expected, performance decreases with higher voxelization resolution since finer details are captured at 64^3. We observe that $\text{IoU}^{\text{Im}}_{\text{3D}} < \text{IoU}^{\text{Can}}_{\text{3D}}$ which is expected since

$\text{IoU}_{\text{3D}}^{\text{Im}}$ captures errors in both pose and 3D shape.

We also evaluate variants of 3DMD by gradually replacing our end-to-end learned components with expert priors. First, we substitute our template learner with an expert template (referred as 3DMD+ in Tab. 6.1), similar to in U-CMR. We find that 3DMD+ slightly improves 3DMD in $\text{IoU}_{\text{3D}}^{\text{Can}}$ (0.705 vs 0.658) and outperforms U-CMR (0.705 vs 0.680). We note a slight degradation in $\text{IoU}_{\text{3D}}^{\text{Im}}$ for 3DMD+ and 3DMD (0.481 vs 0.443). This effect suggests that the template learned in 3DMD, which is encouraged to fit silhouettes from *image-view*, is a better fit for an end-to-end learning approach than an external hand-designed CAD model. To this end, an analysis of the 2D IoU fit on the validation set for our learned template (0.831) and the expert template (0.820) support the above observation, namely that our learned template fits the 2D image-view best.

In addition to the expert template, we also replace our general camera multiplexer with an object-specific camera distribution (referred as 3DMD++ in Tab. 6.1), as in U-CMR. We observe that 3DMD++ performs better compared to our vanilla approach 3DMD for $\text{IoU}_{\text{3D}}^{\text{Can}}$ (0.718 vs 0.705) and more noticeably for $\text{IoU}_{\text{3D}}^{\text{Im}}$ (0.522 vs 0.443). 3DMD++ signifies an upper-bound of our framework when using 3D object priors. We emphasize that 3DMD++, which uses the same supervision as U-CMR, outperforms U-CMR by an appreciable margin (0.718 vs 0.680 $\text{IoU}_{\text{3D}}^{\text{Can}}$).

Unlike prior works [58, 72, 73, 92, 156], we also evaluate our method on all 12 Pascal3D+ categories including aeroplane, bicycle, boat, bottle, bus, car, chair, table, motorbike, sofa, train and TV. We provide our results in Tab. 6.2. We note that objects which have large pose variation, such as aeroplane, under perform objects with less variation, *e.g.* car, bus, sofa, *etc*. We observe objects with high intra-class variation, including chair and table, perform worse than objects with low intra-class variation, *e.g.* bottle. We show qualitative results of learned category templates in Fig. 6.6, and refined meshes in Fig. 6.5. See Suppl. for more results.

Ablations We conduct ablations on the *car* category, focusing on the clusters and our multiplexer.

Figure 6.4: Qualitative 3D mesh reconstructions on the **COCO** dataset for a diverse set of rigid and non-rigid object categories. We visualize input image (row 1), posed template (row 2), and the refined mesh (row $3-5$) in view coordinates from two additional novel viewpoints.

Figure 6.5: Qualitative 3D mesh reconstructions on the **CUB-200** bird (left) and numerous **PASCAL3D+** categories (right). We visualize input image (row 1), posed template (row 2), and the refined mesh (row $3-5$) in view coordinates from two additional novel viewpoints.

Figure 6.6: We show our learned templates (bottom) relative to the canonical pose of their respective reference samples (top).

We use $\text{IoU}^{\text{Im}}_{\text{3D}}$ at resolution of 64^3 in ablation for a finer level of detail.

Cluster Consistency: The cluster consistency loss is designed to encourage the camera poses to be consistent with the image predictor. To understand its effects, we first turn off cluster loss and interaction with the multiplexer. Doing so causes the performance to degrade (0.450 vs 0.408), suggesting cluster consistency is crucial for optimization. We next allow the clusters to learn during training *but* disable their influence in the multiplexer at inference, *e.g.* removing the term L_{KL} from Eq. 6.1. We observe that performance degrades similarly (0.450 vs 0.418), suggesting that cluster consistency is important for the multiplexer. Interestingly, turning off cluster consistency at inference is better than not using cluster consistency in both train and test time (0.418 vs 0.408). Hence, L_{KL} may improve shared features for g, rather than *only* helping with the multiplexer selection.

Cameras Multiplexer: The camera poses used to initialize our multiplexer logically have an impact on performance. We study the effects of using less granular distributions and various priors. First, we observe that utilizing less cameras generally degrades performance. Recall that 3DMD utilizes $N = 48$ cameras spread uniformly over the unit sphere. When we reduce the number of cameras to $N = 32$, we observe an appreciable drop (0.450 vs 0.432). When we reduce the number of

cameras to $N = 16$ the degradation is catastrophic in comparison to the baseline (0.450 vs 0.287).

Following [58], we next spread the camera poses over *only* a single band of azimuth and zero elevation, *e.g*. level to the XZ plane. After training the car category with this configuration we observe an a noticable improvement (0.450 vs 0.482). These results suggest that the car category is slightly biased toward images taken at ground level, which can be beneficial as a prior, at the cost of manual labeling and difficultly in scaling. In contrast, when the aeroplane category uses the constrained camera distribution, the performance degrades greatly as result (0.206 vs 0.152). Hence, although the camera prior distribution is helpful for cars, it is not general for other categories which cover a larger range of rotations, *e.g*. aeroplane, skateboard, baseball bats, tennis rackets among others.

6.4.2 CUB-200

The CUB-200 [158] dataset consist of over 200 bird species with $6{,}033$ images and 2D mask ground truths. Despite not providing 3D mesh ground truths, the dataset has been a popular choice for qualitative evaluation among weakly supervised 3D reconstruction methods [58, 72, 92]. We train our 3DMD on the CUB-200 dataset and show bird reconstructions on the validation set in Fig. 6.5. Since no ground truth shapes are provided, we evaluate the 2D IoU_{2D} between the ground truth bird silhouettes and our rendered predictions. On the validation set, we achieve state-of-the-art performance of 0.792 IoU_{2D} compared to Li et al. [92] that achieve 0.734 and CMR [72] that achieve 0.706.

6.4.3 COCO

The COCO [101] dataset is widely used for 2D object detection and contains about 1.5M object instances across 80 categories. In comparison to Pascal3D+ and CUB-200, the COCO dataset is the most difficult and *in the wild* from the perspective of 3D reconstruction. The annotations tend to be troubled by occlusion, truncation and resultant 2D modal mask and boxes. We select 20 diverse categories: aeroplane, car, bus, train, motorbike, traffic light, stop sign, giraffe, elephant, zebra, bear, skateboard, fire hydrant, tennis racket, baseball bat, umbrella, suitcase, spoon, fork, and knife. We train our approach on the training split and show results on the val set in Fig. 6.4. Additional analysis for our 2D IoU fitting on the COCO dataset is detailed in Suppl.

6.5 Summary

We present a novel framework for weakly supervised 3D mesh discovery in the wild which is able to jointly optimize a 3D template, camera pose, and instance-level meshes. The *only* supervision used is in the form of 2D mask labels. No prior knowledge is imposed. We further present a novel self-supervised cluster consistency loss which encourages the agreement between our camera pose and the image content. We demonstrate the effectiveness of our approach using three datasets Pascal3D+, CUB-200, and COCO. Our method achieves competitive performance on Pascal3D+ while using less 3D expert knowledge compared to prior work (*e.g.*, CAD models or keypoints). We also demonstrate new state-of-the-art results when similar 3D priors *are* assumed. Critically, we generalize to tens of categories by creating a method free of manually set hyperparameters.

6.6 Supplementary Materials

6.6.1 Experimental Results

We provide additional experiments and qualitative results from 3DMD, our 3D mesh reconstruction method, using datasets Pascal3D+ [170], CUB-200 [158], and COCO [101].

2D IoU Performance Tab. 6.3 provides a comprehensive evaluation on the validation set for each category in COCO. We report the performance using our rendered posed template $\mathcal{P}$ and our final refined mesh $\mathcal{M}$, compared to the ground truth segmentation mask. 2D IoU increases for our final mesh prediction compared to the posed template, which is expected as the mesh refinement module is trained to deform the posed template to fit to the object instance in the input image. We observe that IoU is high for rigid categories, such as *stop sign*, *car*, *fire hydrant*, and is lower for categories with thin structures such as *spoon*, *fork* and *knife*.

Qualitative Examples We provide additional examples of our estimated posed template $\mathcal{P}$ and final refined mesh $\mathcal{M}$. We give examples for the car category of Pascal3D+ in Fig. 6.7 and the birds of CUB-200 in Fig. 6.8. Lastly, we provide many other diverse category examples for COCO in Fig. 6.9 and for more Pascal3D+ categories in Fig. 6.10.

Camera Multiplexer Visualizations We provide *comprehensive* visualizations of our camera multiplexer renderings in Fig. 6.11. In order to help build intuition for the multiplexer process, we show all rendered camera views, the predicted clusters and mask, and highlight the resultant best pose i^*.

Category	$\mathcal{P}$ IoU_{2D}	$\mathcal{M}$ IoU_{2D}
Skateboard	0.345	0.478
Spoon	0.396	0.436
Fork	0.400	0.433
Baseball bat	0.490	0.606
Knife	0.497	0.555
Aeroplane	0.536	0.725
Umbrella	0.560	0.717
Bicycle	0.571	0.646
Giraffe	0.603	0.740
Motorbike	0.661	0.723
Zebra	0.670	0.759
Elephant	0.676	0.760
Tennis racket	0.696	0.786
Train	0.696	0.813
Suitcase	0.728	0.793
Truck	0.736	0.787
Fire hydrant	0.742	0.820
Car	0.744	0.831
Bus	0.761	0.800
Traffic light	0.767	0.859
Stop Sign	0.830	0.895
Bear	0.858	0.860

Table 6.3: We show the 2D IoU performance for our posed template $\mathcal{P}$ and the final refined mesh $\mathcal{M}$ across respective validation splits.

6.6.2 Architecture Details

We first detail our overall network architecture in Tab. 6.4, which takes a single input image and outputs a binary segmentation mask, clusters map, camera transformations, and instance-level mesh offsets. 3DMD runs at ≈ 7 frames per second on a NVIDIA 1080 Ti GPU.

Regularization We regularize the template $\mathcal{T}$ and final refined meshes $\mathcal{M}$ using four regularizers with respective loss weights: (i) $\frac{1}{3}$ mean edge length, (ii) $\frac{1}{3}$ standard deviation edge lengths, (iii) $\frac{1}{3}$ laplacian regularizers [37], and (iv) $\frac{1}{10}$ mesh normal consistency. Lastly, we triple the regularization loss weights for the final refined meshes.

Index	Inputs	Operation	Notes	Output shape
(0)	*Input*	$-$	Image	$128 \times 128 \times 3$
(1)	(0)	ResNet18 backbone	Spatial Features	$32 \times 32 \times 512$
(2)	(1)	TConv($512 \rightarrow 256,\ 4 \times 4$), ReLU		$64 \times 64 \times 256$
(3)	(2)	Conv($256 \rightarrow 1,\ 3 \times 3$), Sigmoid	Mask	$64 \times 64 \times 1$
(4)	(1)	TConv($512 \rightarrow 256,\ 4 \times 4$), ReLU		$64 \times 64 \times 256$
(5)	(4)	Conv($256 \rightarrow K,\ 3 \times 3$), Softmax	Clusters	$64 \times 64 \times K$
(6)	(1)	Conv($512 \rightarrow 1,\ 1 \times 1$), Vectorize, Softmax	Attention	1024×1
(7)	(1)	Vectorize		1024×512
(8)	$(6), (7)$	$*,\ +$	Linear Features	512
(9)	(8)	Linear($512 \rightarrow 6N$), Tanh	Rotations	$N \times 6$
(10)	(8)	Linear($512 \rightarrow 3N$), Tanh	Translations	$N \times 3$
(11)	(8)	Linear($512 \rightarrow 3N$), Tanh	Scales	$N \times 3$
(12)	$(1), \mathcal{P}$	Vert alignment		$\lvert\mathcal{V}\rvert \times 512$
(13)	(12)	Linear($512 \rightarrow 256$), BN, ReLU		$\lvert\mathcal{V}\rvert \times 256$
(14)	(13)	Graph conv, BN, ReLU		$\lvert\mathcal{V}\rvert \times 256$
(15)	(14)	Graph conv, BN, ReLU	Vertex Features	$\lvert\mathcal{V}\rvert \times 256$
(16)	(15)	Linear($512 \rightarrow 3$), Tanh	Mesh offsets	$\lvert\mathcal{V}\rvert \times 3$

Table 6.4: We detail our overall network architecture for estimating a binary mask, clusters map, $\Delta\Pi^I$ transformations made of translations & scales in XYZ and $6d$ rotations [190], and lastly refined mesh offsets. TConv denotes transposed convolution and BN denotes batch normalization [70]. We denote N as the number of cameras in our multiplexer while $|\mathcal{V}|$ denotes the number of vertices in the meshes $\mathcal{P}$.

Hyperparameters We define the vertex movement λ_r for the mesh refinement outputs (*e.g.* index 16 in Tab. 6.4) adaptively based on the average IoU_{2D} of the template shape. Formally we define $\lambda_r = 0.75 \cdot (1 - \text{IoU}_{\text{2D}})^{1.5}$. Intuitively, the worse the template mesh fits the *overall* training dataset, the more freedom we give to the mesh refiner. We define the template vertex movement λ_o adaptively based on the 2D mask occupancy of the object category o. We set $\lambda_o = 0.75 \cdot (1 - o)^{\frac{5}{3}}$. Intuitively, we allow categories with a wider variance of 2D masks to deform the template more.

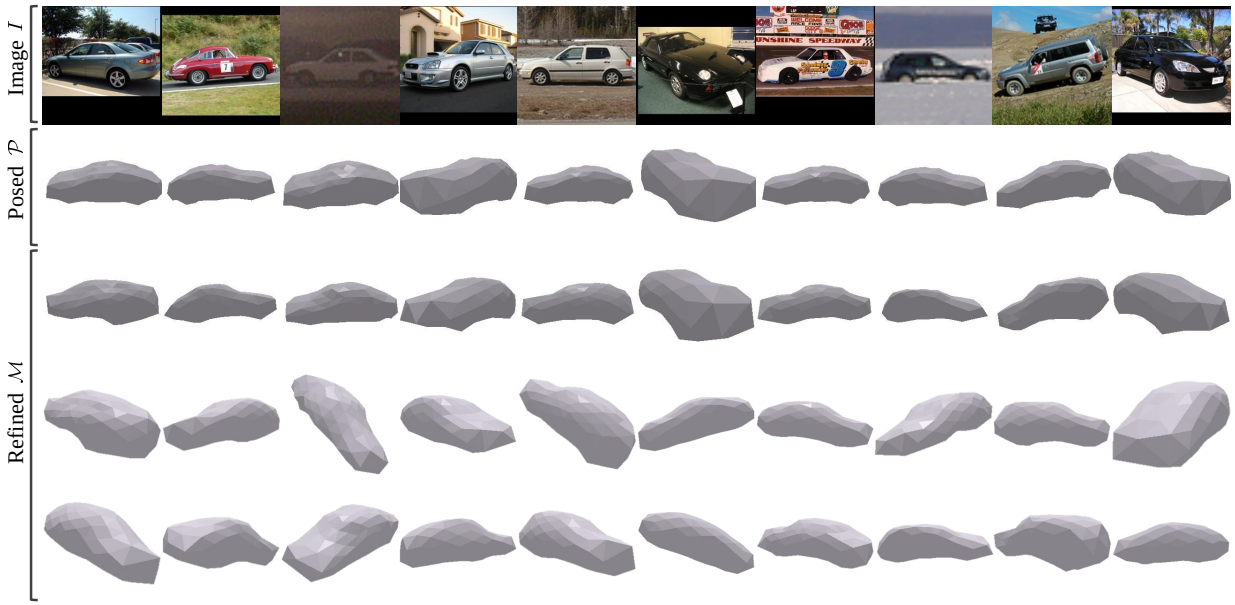

Figure 6.7: Qualitative 3D mesh reconstructions on additional **Pascal3D+** cars. We visualize input image (row 1), posed template (row 2), and the refined mesh (row $3-5$) in view coordinates from two additional novel viewpoints.

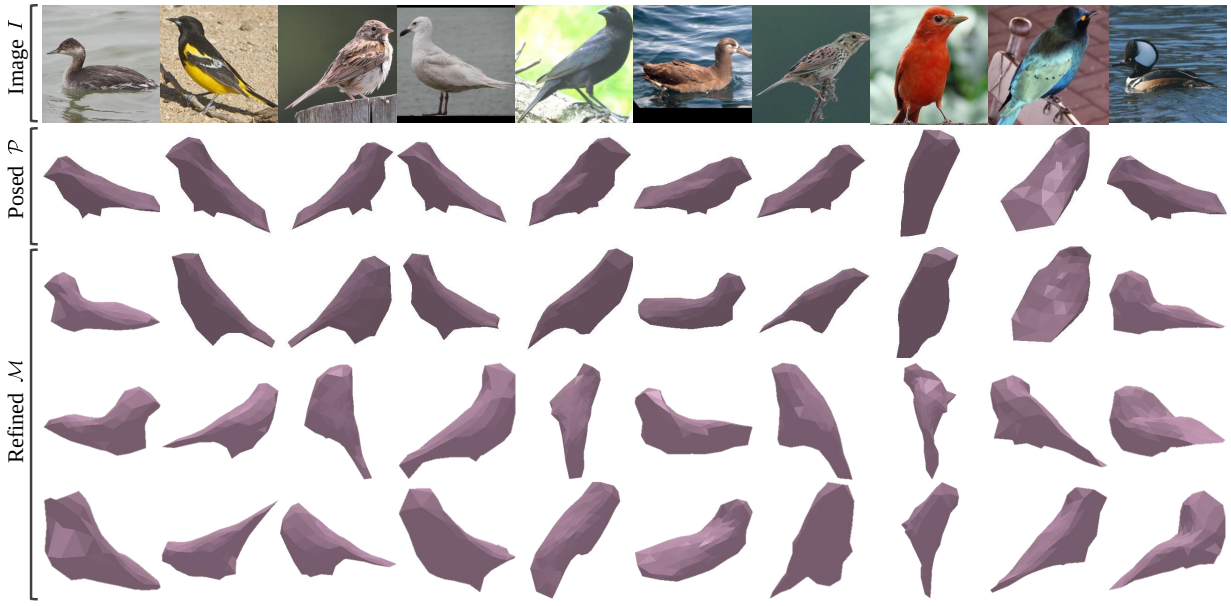

Figure 6.8: Qualitative 3D mesh reconstructions on additional **CUB-200** birds. We visualize input image (row 1), posed template (row 2), and the refined mesh (row $3-5$) in view coordinates from two additional novel viewpoints.

Figure 6.9: Qualitative 3D mesh reconstructions on additional **COCO** categories (right). We visualize input image (row 1), posed template (row 2), and the refined mesh (row $3-5$) in view coordinates from two additional novel viewpoints.

Figure 6.10: Qualitative 3D mesh reconstructions on additional **Pascal3D+** categories. We visualize input image (row 1), posed template (row 2), and the refined mesh (row $3-5$) in view coordinates from two additional novel viewpoints.

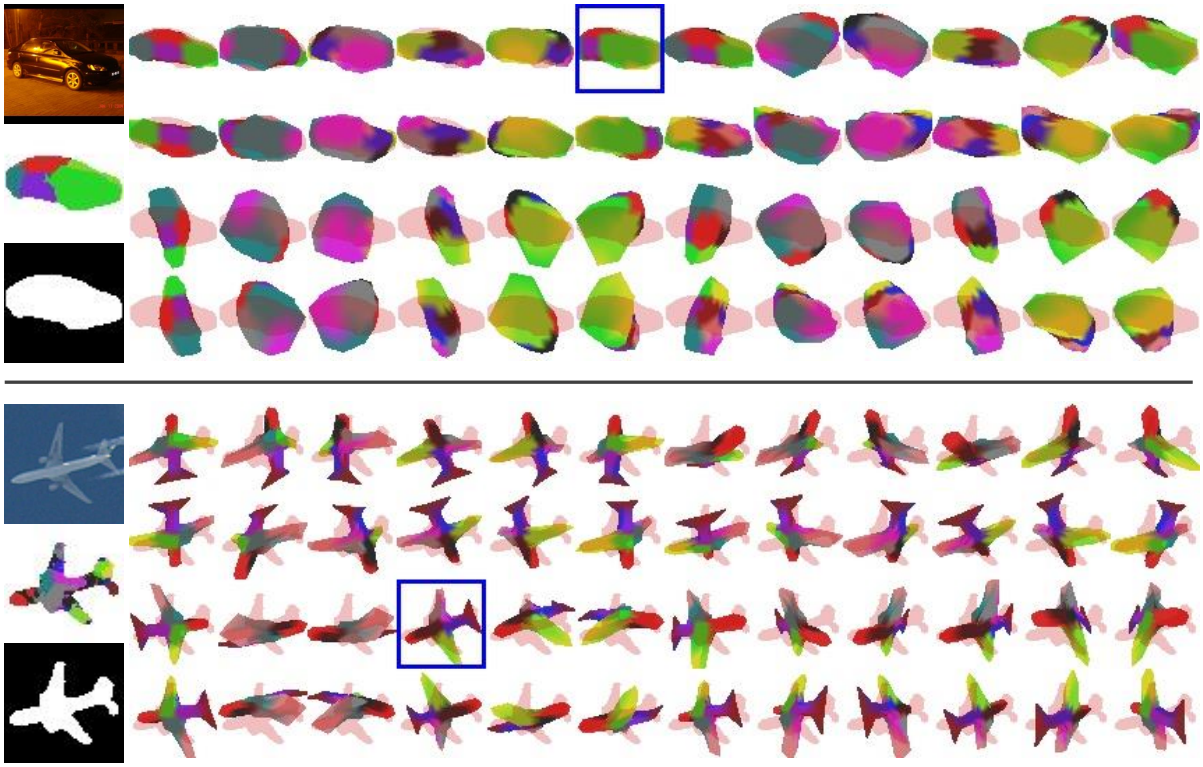

Figure 6.11: We show examples of our camera multiplexer renderings (right) as compared to the network predicted cluster map (left middle) and predicted mask (left bottom) in order to select the best pose i^* highlighted in blue for each example category.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Detecting safety-critical objects in urban scenes using only a single monocular image in both 2D and 3D space is exceedingly challenging due to scale variability, occlusion, and depth ambiguity. The need to detect such objects in an accurate and efficient manner is growing in importance as urban autonomous driving applications and technology continue to rapidly develop. Specifically, monocular camera-based object detection methodology remains the most cost-effective and is therefore widespread among the available sensors (LiDAR, RADAR, and stereo cameras) [16].

Throughout this dissertation, we have proposed and demonstrated the effectiveness of various monocular object detection techniques while keeping both accuracy and efficiency in mind. We apply this goal firstly to pedestrian detection (Ch. 2-3) and secondly for multi-class 3D object detection (Ch. 4-5). We observe from Ch. 2 that the pedestrian detection and semantic segmentation tasks are highly synergistic and can be efficiently paired together during training, without requiring additional data or adding overhead to the runtime speed. In Ch. 3, we further demonstrate how autoregressive phases can be built *into* a single network to further improve accuracy without

sacrificing runtime efficiency. Meanwhile, in Ch. 4 we demonstrate the critical step of extending such object detection systems into 3D space by proposing an efficient single-network 3D region proposal network. We validate our 3D detection system not only on pedestrians but also for cars and cyclists while continuing to use a single model. In Ch. 5 extend our method to video using an efficient 3D Kalman filter and observe more stable accuracy without sacrificing runtime. In Ch. 6, we go beyond perception of 3D cuboids by discovering 3D mesh shapes in cars and $20+$ object categories while using only a single end-to-end model, and *exclusively* 2D supervision.

Our proposed methods stand as modules compatible to a shared core framework derived from the principles of seminal work Faster R-CNN [132]. As a whole, each method explores the value and relative effect of pairing object detection with weak shape supervision, intermediate iterative supervision, 3D supervision, and weakly supervised 3D mesh reconstruction.

7.2 Future Work Suggestions

7.2.1 Maps for 3D Monocular Object Detection

Even before the rapid growth of autonomous driving, maps have been widespread and common in everyday life [117]. The usage of detailed and comprehensive maps is expected to remain a critical component to autonomous driving [12]. Yet, the synergy between *relatively* inexpensive camera-based 3D object detection and maps has not been thoroughly explored. Specifically, maps lanes and elevation-based maps intuitively provide critical constraints to where objects may realistically appear in 3D scenes, which orientation they may be in, *etc*. Although maps are generally fairly expensive to annotate, it is probable to be less expensive to measure a *single* set of maps usable in all vehicles compared to utilizing a high-powered sensor array into all vehicles. Moreover, increased automatic methods for map generation are being studied in parallel [61]. Hence, careful

and efficient integration of critical map data into monocular 3D object detection systems could provide substantial benefits to accuracy and robustness for known driving terrain.

7.2.2 Monocular 3D Object Detection with Mesh Discovery

A natural extension of monocular 3D object detection (Ch. 4-5) and weakly supervised mesh reconstruction (Ch. 6) is their joining for a total comprehensive full-scene understanding. The combination of such two problems would result in weakly supervised joint 3D object localization and reconstruction — a probably absurdly ambitious task. The most challenging aspects of this direction are technological. Full-scene rendering is both progromatically difficult and likely expensive for consumed GPU memory. A soft variant of this idea is partially explored for autolabeling 3D cuboids [181], with the added caveat of requiring LiDAR. It is worth emphasizing that object reconstructions (Ch. 6 and [58, 72, 92]) have significant room for improvement even with clean 2D detections. Naturally, the pairing of these two ideas has a lot of room for innovation and is attractive from the perspective of training a *full* 3D recognition system with modular amounts of 3D supervision, *e.g.*, $\pm$ 3D localization, $\pm$ 3D cuboid shape, and $\pm$ object pose. The most extreme and exciting setting being **full scene 3D localization and reconstruction from 2D supervision**.

APPENDIX

APPENDIX

A.1 Publications

A list of all peer-reviewed publications during the MSU PhD program listed chronologically.

- *Garrick Brazil*, Xi Yin, and Xiaoming Liu. "Illuminating pedestrians via simultaneous detection & segmentation." Proceedings of the IEEE International Conference on Computer Vision. 2017.

- Terwilliger, Adam, *Garrick Brazil*, and Xiaoming Liu. "Recurrent flow-guided semantic forecasting." 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). IEEE, 2019.

- *Garrick Brazil*, and Xiaoming Liu. "Pedestrian detection with autoregressive network phases." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.

- *Garrick Brazil*, and Xiaoming Liu. "M3D-RPN: Monocular 3D region proposal network for object detection." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019. **(Oral Top** 4.3%**)**

- Shengjie Zhu, *Garrick Brazil*, and Xiaoming Liu. "The edge of depth: Explicit constraints between segmentation and depth." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

- *Garrick Brazil*, Gerard Pons-Moll, Xiaoming Liu, Bernt Schiele. "Kinematic 3d object detection in monocular video." European Conference on Computer Vision. Springer, Cham, 2020.

- Abhinav Kumar, *Garrick Brazil*, Xiaoming Liu. "GrooMeD-NMS: Grouped Mathematically Differentiable NMS for Monocular 3D Object Detection." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

- *Garrick Brazil*, Georgia Gkioxari, Nikhila Ravi, Justin Johnson, Xiaoming Liu. "3D Mesh Discovery in the Wild" **Under review** in the IEEE/CVF International Conference on Computer Vision. 2021.

A.2 Video Demonstrations

- Illuminating Pedestrians via Simultaneous Detection & Segmentation `https://youtu.be/9WlC4Qff3mk`

- Pedestrian Detection with Autoregressive Network Phases `https://youtu.be/FIgTXQVGUHQ`

- M3D-RPN: Monocular 3D Region Proposal Network for Object Detection `https://youtu.be/qu7YMMjnUAk`

- Kinematic 3D Object Detection in Monocular Video `https://youtu.be/PRmYzHtQ99M`

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Y. Almalioglu, M. R. U. Saputra, P. P. de Gusmao, A. Markham, and N. Trigoni. Ganvo: Unsupervised deep monocular visual odometry and depth estimation with generative adversarial networks. In *International Conference on Robotics and Automation (ICRA)*, pages 5474–5480. IEEE, 2019. 105

[2] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014. 9, 12

[3] Y. Atoum, J. Roth, M. Bliss, W. Zhang, and X. Liu. Monocular video-based trailer coupler detection using multiplexer convolutional neural network. In *International Conference on Computer Vision (ICCV)*. IEEE, 2017. 59

[4] A. Badki, A. Troccoli, K. Kim, J. Kautz, P. Sen, and O. Gallo. Bi3d: Stereo depth estimation via binary classifications. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1600–1608, 2020. 105

[5] A. Behl, O. Hosseini Jafari, S. Karthik Mustikovela, H. Abu Alhaija, C. Rother, and A. Geiger. Bounding boxes, segmentations and object coordinates: How important is recognition for 3D scene flow estimation in autonomous driving scenarios? In *International Conference on Computer Vision (ICCV)*. IEEE, 2017. 55

[6] G. Bertasius, L. Torresani, and J. Shi. Object detection in video with spatiotemporal sampling networks. In *European Conference on Computer Vision (ECCV)*. Springer, 2018. 82

[7] H. Bertiche, M. Madadi, and S. Escalera. Cloth3d: Clothed 3D humans. In *European Conference on Computer Vision (ECCV)*, pages 344–359. Springer, 2020. 104

[8] B. L. Bhatnagar, G. Tiwari, C. Theobalt, and G. Pons-Moll. Multi-garment net: Learning to dress 3D people from images. In *International Conference on Computer Vision (ICCV)*, pages 5420–5430, 2019. 104

[9] G. Brazil and X. Liu. M3D-RPN: Monocular 3D region proposal network for object detection. In *International Conference on Computer Vision (ICCV)*. IEEE, 2019. xvi, xvii, 77, 80, 83, 84, 91, 92, 94, 95, 96, 98, 102

[10] G. Brazil and X. Liu. Pedestrian detection with autoregressive network phases. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019. 57

[11] G. Brazil, X. Yin, and X. Liu. Illuminating pedestrians via simultaneous detection segmentation. In *International Conference on Computer Vision (ICCV)*. IEEE, 2017. 33, 36, 43, 44, 45, 46, 47, 48, 57, 77

[12] D. J. Burnette, A. H. Chatham, and M. P. McNaughton. Automatic collection of quality control statistics for maps used in autonomous driving, Sept. 3 2013. US Patent 8,527,199. 131

[13] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision (ECCV)*. Springer, 2016. 1, 9, 10, 11, 12, 14, 23, 24, 45, 47, 48, 55, 57

[14] Z. Cai, M. Saberian, and N. Vasconcelos. Learning complexity-aware cascades for deep pedestrian detection. In *International Conference on Computer Vision (ICCV)*. IEEE, 2015. 24

[15] Z. Cai and N. Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 36, 55, 57

[16] S. Campbell, N. O'Mahony, L. Krpalcova, D. Riordan, J. Walsh, A. Murphy, and C. Ryan. Sensor technology in autonomous vehicles: a review. In *2018 29th Irish Signals and Systems Conference (ISSC)*, pages 1–4. IEEE, 2018. 2, 130

[17] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. 37

[18] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere, and T. Chateau. Deep MANTA: A coarse-to-fine many-task network for joint 2D and 3D vehicle analysis from monocular image. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. 81, 82

[19] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015. 107

[20] J.-R. Chang and Y.-S. Chen. Pyramid stereo matching network. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 80

[21] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *Pattern Analysis and Machine Intelligence (PAMI)*, 40(4):834–848, 2017. 9

[22] W. Chen, H. Ling, J. Gao, E. Smith, J. Lehtinen, A. Jacobson, and S. Fidler. Learning to predict 3D objects with an interpolation-based differentiable renderer. In *Neural Information Processing Systems (NeurIPS)*, 2019. 105, 107

[23] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3D object detection for autonomous driving. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. 6, 55, 59, 60, 66, 70, 71, 72, 75, 80

[24] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun. 3D object proposals for accurate object class detection. In *Neural Information Processing Systems (NeurIPS)*, pages 424–432, 2015. xii, xvi, 6, 55, 59, 60, 66, 69, 70, 71, 72, 74, 75, 80, 93, 94, 95, 96, 100, 102

[25] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3D object detection network for autonomous driving. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. 58

[26] Y. Chen, S. Liu, X. Shen, and J. Jia. Fast point R-CNN. In *International Conference on Computer Vision (ICCV)*. IEEE, 2019. 77

[27] Y. Chen, J. Wang, J. Li, C. Lu, Z. Luo, H. Xue, and C. Wang. LiDAR-video driving dataset: Learning driving policies effectively. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 55

[28] Z. Chen, S. Huang, and D. Tao. Context refinement for object detection. In *European Conference on Computer Vision (ECCV)*. Springer, 2018. 55

[29] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3D-r2n2: A unified approach for single and multi-view 3D object reconstruction. In *European Conference on Computer Vision (ECCV)*, 2016. 107

[30] H. Chu, W.-C. Ma, K. Kundu, R. Urtasun, and S. Fidler. Surfconv: Bridging 3D and 2D convolution for RGBD images. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 55, 58

[31] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. xiv, 9, 10, 12, 20, 82

[32] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. 10, 12

[33] Q. Dai, V. Patil, S. Hecker, D. Dai, L. Van Gool, and K. Schindler. Self-supervised object motion and depth estimation from video. In *Computer Vision and Pattern Recognition Workshop (CVPRW)*, pages 1004–1005, 2020. 105

[34] A. Dame, V. A. Prisacariu, C. Y. Ren, and I. Reid. Dense reconstruction using 3D object shape priors. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1288–1295, 2013. 104

[35] M. de La Gorce, D. J. Fleet, and N. Paragios. Model-based 3D hand pose estimation from monocular video. *Pattern Analysis and Machine Intelligence (PAMI)*, 2011. 105

[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale

hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009. 15

[37] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Special Interest Group on Computer Graphics (SIGGRAPH)*, pages 317–324, 1999. 125

[38] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009. xv, 3, 10, 12, 13, 14, 18, 20, 21, 22, 24, 25, 33, 35, 46, 47

[39] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *Pattern Analysis and Machine Intelligence (PAMI)*, 34(4):743–761, 2012. 12, 22, 46, 47, 77

[40] X. Du, M. H. Ang, S. Karaman, and D. Rus. A general pipeline for 3D detection of vehicles. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2018. 58

[41] X. Du, M. El-Khamy, J. Lee, and L. Davis. Fused DNN: A deep neural network fusion approach to fast and robust pedestrian detection. In *Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017. 12, 13, 14, 24, 33, 36, 45, 47

[42] B. A. Erol, A. Majumdar, J. Lwowski, P. Benavidez, P. Rad, and M. Jamshidi. Improved deep neural network object tracking system for applications in home robotics. In *Computational Intelligence for Pattern Recognition*, pages 369–395. Springer, 2018. 1

[43] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html. 23

[44] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 11

[45] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3D object reconstruction from a single image. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 107

[46] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence (PAMI)*, 32(9):1627–1645, 2010. 80

[47] S. Fidler, R. Mottaghi, A. Yuille, and R. Urtasun. Bottom-up segmentation for top-down detection. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2013. 9, 13

[48] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. DSSD: Deconvolutional single shot

detector. *arXiv preprint arXiv:1701.06659*, 2017. 45

[49] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 80

[50] M. Gao, R. Yu, A. Li, V. I. Morariu, and L. S. Davis. Dynamic zoom-in network for fast object detection in large images. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 55

[51] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012. xi, 1, 3, 9, 10, 13, 20, 21, 23, 35, 45, 46, 55, 57, 62, 69, 70, 78, 79, 82, 93, 94, 100

[52] S. Giancola, J. Zarzar, and B. Ghanem. Leveraging shape completion for 3D siamese tracking. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019. 82

[53] S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *International Conference on Computer Vision (ICCV)*. IEEE, 2015. 11

[54] R. Girshick. Fast R-CNN. In *International Conference on Computer Vision (ICCV)*. IEEE, 2015. 11, 12, 15, 41, 43, 64, 72

[55] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014. 12

[56] G. Gkioxari, J. Malik, and J. Johnson. Mesh r-cnn. In *International Conference on Computer Vision (ICCV)*, 2019. 105, 107, 114, 115

[57] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow. Digging into self-supervised monocular depth estimation. In *International Conference on Computer Vision (ICCV)*, pages 3828–3838, 2019. 105

[58] S. Goel, A. Kanazawa, and J. Malik. Shape and viewpoint without keypoints. In *European Conference on Computer Vision (ECCV)*, 2020. xiii, 105, 106, 107, 108, 109, 110, 111, 112, 117, 118, 119, 122, 132

[59] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Neural Information Processing Systems (NeurIPS)*, pages 545–552, 2009. 37

[60] J. Guerry, A. Boulch, B. Le Saux, J. Moras, A. Plyer, and D. Filliat. SnapNet-R: Consistent 3D multi-view semantic labeling for robotics. In *International Conference on Computer*

Vision (ICCV). IEEE, 2017. 55

[61] C. Guo, K. Kidono, J. Meguro, Y. Kojima, M. Ogawa, and T. Naito. A low-cost solution for automatic lane-level map generation using conventional in-car sensors. *Transactions on Intelligent Transportation Systems (T-ITS)*, 17(8):2355–2366, 2016. 131

[62] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision (ECCV)*. Springer, 2014. 9, 11, 12

[63] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 105

[64] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *International Conference on Computer Vision (ICCV)*, 2017. 105

[65] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. 11, 12, 39, 105

[66] H.-N. Hu, Q.-Z. Cai, D. Wang, J. Lin, M. Sun, P. Krahenbuhl, T. Darrell, and F. Yu. Joint monocular 3D vehicle detection and tracking. In *International Conference on Computer Vision (ICCV)*. IEEE, 2019. 82

[67] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. 66, 93

[68] W.-C. Hung, V. Jampani, S. Liu, P. Molchanov, M.-H. Yang, and J. Kautz. Scops: Self-supervised co-part segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, pages 869–878, 2019. 108

[69] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 25

[70] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015. xiii, 40, 126

[71] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 15, 18, 45

[72] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning category-specific mesh reconstruction from image collections. In *European Conference on Computer Vision (ECCV)*,

2018. xiii, 105, 107, 108, 109, 112, 117, 118, 119, 122, 132

[73] A. Kar, S. Tulsiani, J. Carreira, and J. Malik. Category-specific object reconstruction from a single image. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1966–1974, 2015. xiii, 107, 109, 117, 118, 119

[74] H. Kato, Y. Ushiku, and T. Harada. Neural 3D mesh renderer. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 105

[75] J. N. S. L. B. Kim and G. Kim. Improving occlusion and hard negative handling for single-stage pedestrian detectors. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 33, 45, 48

[76] S.-W. Kim, H.-K. Kook, J.-Y. Sun, M.-C. Kang, and S.-J. Ko. Parallel feature pyramid network for object detection. In *European Conference on Computer Vision (ECCV)*. Springer, 2018. 2, 55

[77] A. Kirillov, Y. Wu, K. He, and R. Girshick. Pointrend: Image segmentation as rendering. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 105

[78] T. Kong, F. Sun, W. Huang, and H. Liu. Deep feature pyramid reconfiguration for object detection. In *European Conference on Computer Vision (ECCV)*. Springer, 2018. 34, 37, 49, 57

[79] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems (NeurIPS)*, pages 1097–1105, 2012. 25, 56, 64, 105

[80] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. Joint 3D proposal generation and object detection from view aggregation. In *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018. 80

[81] J. Ku, A. D. Pon, and S. L. Waslander. Monocular 3D object detection leveraging accurate proposals and shape reconstruction. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019. xii, 77, 80, 87, 94, 95, 96, 100, 102, 105

[82] N. Kulkarni, A. Gupta, D. Fouhey, and S. Tulsiani. Articulation-aware canonical surface mapping. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 108, 109, 112

[83] N. Kulkarni, A. Gupta, and S. Tulsiani. Canonical surface mapping via geometric cycle consistency. In *International Conference on Computer Vision (ICCV)*, 2019. 105, 106, 110

[84] A. Kumar, T. K. Marks, W. Mou, Y. Wang, M. Jones, A. Cherian, T. Koike-Akino, X. Liu, and C. Feng. LUVLi face alignment: Estimating landmarks' location, uncertainty, and visibility likelihood. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020. 81

[85] K. S. Kumar, S. Prasad, P. K. Saroj, and R. C. Tripathi. Multiple cameras using real time object tracking for surveillance and security system. In *International Conference on Emerging Trends in Engineering and Technology (IConETech)*, pages 213–218. IEEE, 2010. 1

[86] S. Kumar, Y. Dai, and H. Li. Monocular dense 3D reconstruction of a complex dynamic scene from two perspective frames. In *International Conference on Computer Vision (ICCV)*, pages 4649–4657, 2017. 104

[87] W. Kuo, A. Angelova, T.-Y. Lin, and A. Dai. Mask2cad: 3D shape prediction by learning to segment and retrieve. In *European Conference on Computer Vision (ECCV)*, 2020. 105

[88] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. PointPillars: Fast encoders for object detection from point clouds. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019. 77

[89] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999. 56, 64

[90] B. Li, W. Ouyang, L. Sheng, X. Zeng, and X. Wang. GS3D: An efficient 3D object detection framework for autonomous driving. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019. xii, 77, 80, 87, 94, 95, 96, 100, 102

[91] J. Li, X. Liang, S. Shen, T. Xu, J. Feng, and S. Yan. Scale-aware fast R-CNN for pedestrian detection. *arXiv preprint arXiv:1510.08160*, 2015. 10, 11, 24, 48

[92] X. Li, S. Liu, K. Kim, S. De Mello, V. Jampani, M.-H. Yang, and J. Kautz. Self-supervised single-view 3D reconstruction via semantic consistency. In *European Conference on Computer Vision (ECCV)*, 2020. xiii, 107, 108, 109, 112, 117, 118, 119, 122, 132

[93] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun. DetNet: Design backbone for object detection. In *European Conference on Computer Vision (ECCV)*. Springer, 2018. 57

[94] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun. Multi-task multi-sensor fusion for 3D object detection. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019. 77

[95] M. Liang, B. Yang, S. Wang, and R. Urtasun. Deep continuous fusion for multi-sensor 3D object detection. In *European Conference on Computer Vision (ECCV)*. Springer, 2018. 55, 58, 77

[96] C. Lin, J. Lu, G. Wang, and J. Zhou. Graininess-aware deep feature learning for pedestrian detection. In *European Conference on Computer Vision (ECCV)*. Springer, 2018. 45, 48

[97] C.-H. Lin, C. Kong, and S. Lucey. Learning efficient point cloud generation for dense 3D object reconstruction. In *Association for the Advancement of Artificial Intelligence Confer-*

ence on Artificial Intelligence (AAAI), 2018. 107

[98] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. 1, 34, 37, 39, 49

[99] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. 1

[100] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*. Springer, 2014. 10

[101] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014. 105, 106, 107, 109, 116, 123, 124

[102] F. Liu and X. Liu. Learning implicit functions for topology-varying dense 3D shape correspondence. In *Neural Information Processing Systems (NeurIPS)*, 2020. 107

[103] L. Liu, J. Lu, C. Xu, Q. Tian, and J. Zhou. Deep fitting degree scoring network for monocular 3D object detection. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019. xii, 77, 80, 81, 87, 94, 95, 96, 100, 102

[104] M. Liu and M. Zhu. Mobile video object detection with temporally-aware feature maps. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 82

[105] S. Liu, W. Chen, T. Li, and H. Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. In *International Conference on Computer Vision (ICCV)*, 2019. 105

[106] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 34, 37, 49

[107] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*. Springer, 2016. 57

[108] W. Liu, S. Liao, W. Hu, X. Liang, and X. Chen. Learning efficient single-stage pedestrian detectors by asymptotic localization fitting. In *European Conference on Computer Vision (ECCV)*. Springer, 2018. 33, 37, 44, 45, 57

[109] X. Liu and T. Yu. Gradient feature selection for online boosting. In *International Conference on Computer Vision (ICCV)*. IEEE, 2007. 57

[110] S. Lloyd. Least squares quantization in pcm. *Transactions of Information Theory*, 1982. 113

[111] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 105

[112] M. M. Loper and M. J. Black. OpenDR: An approximate differentiable renderer. In *European Conference on Computer Vision (ECCV)*, 2014. 105

[113] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, and X. Fan. Accurate monocular 3D object detection via color-embedded 3d reconstruction for autonomous driving. In *International Conference on Computer Vision (ICCV)*. IEEE, 2019. 77, 80, 94, 95, 102, 105

[114] F. Manhardt, W. Kehl, and A. Gaidon. ROI-10D: Monocular lifting of 2D detection to 6D pose and metric shape. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019. xii, 77, 80, 87, 94, 95, 96, 100, 102

[115] B. McCann, J. Bradbury, C. Xiong, and R. Socher. Learned in translation: Contextualized word vectors. In *Neural Information Processing Systems (NeurIPS)*, 2017. 37

[116] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 107

[117] C. C. Miller. A beast in the field: The google maps mashup as gis/2. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 41(3):187–199, 2006. 131

[118] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3D bounding box estimation using deep learning and geometry. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. xvi, 6, 55, 58, 59, 60, 66, 70, 71, 72, 75, 80

[119] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision (ECCV)*. Springer, 2016. 34, 37, 49

[120] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt, and Y.-L. Yang. Hologan: Unsupervised learning of 3D representations from natural images. In *International Conference on Computer Vision (ICCV)*, Nov 2019. 108

[121] D. Novotny, N. Ravi, B. Graham, N. Neverova, and A. Vedaldi. C3dpo: Canonical 3D pose networks for non-rigid structure from motion. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 105

[122] K. Oksuz, B. Can Cam, E. Akbas, and S. Kalkan. Localization recall precision (LRP): A new performance metric for object detection. In *European Conference on Computer Vision*

(ECCV). Springer, 2018. 57

[123] W. Ouyang, K. Wang, X. Zhu, and X. Wang. Chained cascade network for object detection. In *International Conference on Computer Vision (ICCV)*. IEEE, 2017. 36

[124] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 107

[125] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017. 65, 68, 75, 92

[126] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3D object detection from RGB-D data. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 55, 58, 80

[127] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Neural Information Processing Systems (NeurIPS)*, 2017. 80

[128] H. Qin, J. Yan, X. Li, and X. Hu. Joint training of cascaded CNN for face detection. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. 36

[129] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari. Accelerating 3D deep learning with pytorch3d. In *Special Interest Group on Computer Graphics (SIGGRAPH)*, 2020. 105, 107, 116

[130] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. 1, 57, 79, 81, 82

[131] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu. Accurate single stage detector using recurrent rolling convolution. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. 33, 37, 45, 55, 57, 82

[132] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NeurIPS)*, 2015. 2, 5, 10, 11, 14, 38, 43, 45, 57, 61, 63, 64, 72, 79, 80, 81, 82, 83, 84, 131

[133] J. Rock, T. Gupta, J. Thorsen, J. Gwak, D. Shin, and D. Hoiem. Completing 3D object shape from one depth image. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 107

[134] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer Assisted Interventions (MICCAI)*. Springer, 2015. 37, 49

[135] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 82

[136] C. Russell, R. Yu, and L. Agapito. Video pop-up: Monocular 3D reconstruction of dynamic scenes. In *European Conference on Computer Vision (ECCV)*, pages 583–598. Springer, 2014. 104

[137] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *Pattern Analysis and Machine Intelligence (PAMI)*, 39(11):2298–2304, 2017. 37

[138] S. Shi, X. Wang, and H. Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 55, 58

[139] S. Shi, X. Wang, and H. Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019. 77

[140] A. Simonelli, S. R. Bulo, L. Porzi, M. López-Antequera, and P. Kontschieder. Disentangling monocular 3D object detection. In *International Conference on Computer Vision (ICCV)*. IEEE, 2019. xii, 77, 80, 81, 93, 94, 95, 96, 102

[141] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 14, 39, 42, 49, 105

[142] M. Slavcheva, M. Baust, and S. Ilic. Towards implicit correspondence in signed distance field evolution. In *International Conference on Computer Vision Workshop (ICCVW)*, 2017. 107

[143] E. Smith, S. Fujimoto, and D. Meger. Multi-view silhouette and depth decomposition for high resolution 3D object representation. In *Neural Information Processing Systems (NeurIPS)*, 2018. 107

[144] E. J. Smith, S. Fujimoto, A. Romero, and D. Meger. Geometrics: Exploiting geometric structure for graph-encoded objects. In *International Conference on Machine Learning (ICML)*, 2019. 107

[145] N. Smolyanskiy, A. Kamenev, and S. Birchfield. On the importance of stereo for accurate depth estimation: An efficient semi-supervised deep neural network approach. In *Computer Vision and Pattern Recognition Workshop (CVPRW)*, pages 1007–1015, 2018. 105

[146] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI)*, 2017. 37

[147] T. Song, L. Sun, D. Xie, H. Sun, and S. Pu. Small-scale pedestrian detection based on somatic topology localization and temporal feature aggregation. In *European Conference on Computer Vision (ECCV)*. Springer, 2018. 33, 45

[148] R. Stewart, M. Andriluka, and A. Y. Ng. End-to-end people detection in crowded scenes. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. 33, 37

[149] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman. Pix3d: Dataset and methods for single-image 3D shape modeling. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 107

[150] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015. 12, 33, 36

[151] K. Tateno, F. Tombari, I. Laina, and N. Navab. CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. 55

[152] Y. Tian, P. Luo, X. Wang, and X. Tang. Deep learning strong parts for pedestrian detection. In *International Conference on Computer Vision (ICCV)*. IEEE, 2015. 11, 24

[153] F. Tosi, F. Aleotti, M. Poggi, and S. Mattoccia. Learning monocular depth estimation infusing traditional stereo knowledge. In *Computer Vision and Pattern Recognition (CVPR)*, pages 9799–9809, 2019. 105

[154] S. Tsogkas, I. Kokkinos, G. Papandreou, and A. Vedaldi. Deep learning for semantic part segmentation with high-level guidance. *arXiv preprint arXiv:1505.02438*, 2015. 14

[155] S. Tulsiani, A. A. Efros, and J. Malik. Multi-view consistency as supervisory signal for learning shape and pose prediction. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2897–2905, 2018. 107

[156] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2626–2634, 2017. xiii, 107, 109, 117, 118, 119

[157] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu. WaveNet: A generative model for raw audio. In *Speech Synthesis Workshop (SSW)*, 2016. 34

[158] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 106, 108, 116, 122, 124

[159] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. Pixel2mesh: Generating 3D mesh

models from single rgb images. In *European Conference on Computer Vision (ECCV)*, pages 52–67, 2018. 107, 114

[160] X. Wang, T. Xiao, Y. Jiang, S. Shao, J. Sun, and C. Shen. Repulsion loss: Detecting pedestrians in a crowd. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 33, 45, 48, 64

[161] X. Wang, W. Yin, T. Kong, Y. Jiang, L. Li, and C. Shen. Task-aware monocular depth estimation for 3D object detection. In *Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI)*, 2020. 80

[162] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger. Pseudo-LiDAR from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019. 77, 80

[163] Y. Wang, P. Wang, Z. Yang, C. Luo, Y. Yang, and W. Xu. Unos: Unified unsupervised optical-flow and stereo-depth estimation by watching videos. In *Computer Vision and Pattern Recognition (CVPR)*, pages 8071–8081, 2019. 105

[164] G. Welch, G. Bishop, et al. An introduction to the kalman filter. 1995. 78, 83, 89

[165] C.-Y. Weng, B. Curless, and I. Kemelmacher-Shlizerman. Photo wake-up: 3D character animation from a single photo. In *Computer Vision and Pattern Recognition (CVPR)*, June 2019. 104

[166] A. Womg, M. J. Shafiee, F. Li, and B. Chwyl. Tiny SSD: A tiny single-shot detection deep convolutional neural network for real-time embedded object detection. In *Conference on Robots and Vision (CRV)*. IEEE, 2018. 2, 57

[167] J. Wu, Y. Wang, T. Xue, X. Sun, W. T. Freeman, and J. B. Tenenbaum. MarrNet: 3D Shape Reconstruction via 2.5D Sketches. In *Neural Information Processing Systems (NeurIPS)*, 2017. 107

[168] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *Neural Information Processing Systems (NeurIPS)*, 2016. 105, 107

[169] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Subcategory-aware convolutional neural networks for object proposals and detection. In *Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017. 69, 70

[170] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond pascal: A benchmark for 3D object detection in the wild. In *Winter Conference on Applications of Computer Vision (WACV)*, 2014. 106, 107, 108, 116, 124

[171] F. Xiao and Y. Jae Lee. Video object detection with an aligned spatial-temporal memory. In *European Conference on Computer Vision (ECCV)*. Springer, 2018. 82

[172] B. Xu and Z. Chen. Multi-level fusion based 3D object detection from monocular images. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. xvi, 6, 55, 58, 59, 60, 66, 70, 71, 72, 75, 80

[173] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee. Perspective transformer nets: Learning single-view 3D object reconstruction without 3D supervision. In *Neural Information Processing Systems (NeurIPS)*, 2016. 107

[174] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3D object detection from point clouds. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 55, 58

[175] F. Yang, W. Choi, and Y. Lin. Exploit all the layers: Fast and accurate CNN object detector with scale dependent pooling and cascaded rejection classifiers. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. 11, 57, 81, 82

[176] S. Yang and S. Scherer. CubeSLAM: Monocular 3D object SLAM. *Transactions on Robotics (T-RO)*, 35(4):925–938, 2019. 104

[177] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia. STD: Sparse-to-dense 3D object detector for point cloud. In *International Conference on Computer Vision (ICCV)*. IEEE, 2019. 77, 82

[178] Y. Ye, S. Tulsiani, and A. Gupta. Shelf-supervised mesh prediction in the wild. 2021. 108

[179] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang. Unitbox: An advanced object detection network. In *Internet Measurement Conference (ICM)*. ACM, 2016. 64

[180] S. Zagoruyko, A. Lerer, T.-Y. Lin, P. O. Pinheiro, S. Gross, S. Chintala, and P. Dollár. A multipath network for object detection. In *British Machine Vision Conference (BMVC)*, 2016. 33, 36

[181] S. Zakharov, W. Kehl, A. Bhargava, and A. Gaidon. Autolabeling 3d objects with differentiable rendering of sdf shape priors. In *Computer Vision and Pattern Recognition (CVPR)*, pages 12224–12233, 2020. 132

[182] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *Computer Vision and Pattern Recognition (CVPR)*, pages 340–349, 2018. 105

[183] L. Zhang, L. Lin, X. Liang, and K. He. Is faster R-CNN doing well for pedestrian detection? In *European Conference on Computer Vision (ECCV)*. Springer, 2016. 5, 9, 10, 11, 12, 14, 16, 23, 24, 38, 41, 45, 47, 48

[184] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele. How far are we from solving pedestrian detection? In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. 2, 9, 11, 12, 35, 47

[185] S. Zhang, R. Benenson, and B. Schiele. Filtered channel features for pedestrian detection. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015. 22

[186] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li. Occlusion-aware R-CNN: Detecting pedestrians in a crowd. In *European Conference on Computer Vision (ECCV)*. Springer, 2018. 33

[187] S. Zhang, J. Yang, and B. Schiele. Occluded pedestrian detection through guided attention in CNNs. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 33, 45

[188] Z. Zhang, S. Qiao, C. Xie, W. Shen, B. Wang, and A. L. Yuille. Single-shot object detection with enriched semantics. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 2, 57

[189] C. Zhou and J. Yuan. Bi-box regression for pedestrian detection and occlusion estimation. In *European Conference on Computer Vision (ECCV)*. Springer, 2018. 33, 45, 48, 57

[190] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019. xiii, 81, 126

[191] X. Zhu, J. Dai, L. Yuan, and Y. Wei. Towards high performance video object detection. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 82

[192] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei. Flow-guided feature aggregation for video object detection. In *International Conference on Computer Vision (ICCV)*. IEEE, 2017. 82