MonoLayout: Amodal scene layout from a single image

Kaustubh Mani * [1], Swapnil Daga[1], Shubhika Garg[2], N. Sai Shankar[1], J. Krishna Murthy[3,4], and K. Madhava Krishna[1]

[1]Robotics Research Center, KCIS, IIIT Hyderabad, India, [2]IIT Kharagpur, [3]Mila - Quebec AI Institute, Montreal, Canada, [4]Université de Montréal

Figure 1: ***MonoLayout***: Given only a single image of a road scene, we propose a neural network architecture that reasons about the *amodal* scene layout in bird's eye view in *real-time* ($> 30fps$). Our approach, dubbed *MonoLayout* can *hallucinate* regions of the static scene (road, sidewalks)—and traffic participants—that do not even project to the visible regime of the image plane. Shown above are example images from the KITTI [8] (left) and Argoverse [3] (right) datasets. *MonoLayout* outperforms prior art (by more than a 20% margin) on hallucinating occluded regions.

Abstract

In this paper, we address the novel, highly challenging problem of estimating the layout of a complex urban driving scenario. Given a single color image captured from a driving platform, we aim to predict the bird's eye view layout of the road and other traffic participants. The estimated layout should reason beyond what is visible in the image, and compensate for the loss of 3D information due to projection. We dub this problem amodal scene layout estimation*, which involves* hallucinating *scene layout for even parts of the world that are occluded in the image. To this end, we present* MonoLayout*, a deep neural network for real-time amodal scene layout estimation from a single image. We represent scene layout as a multi-channel semantic occupancy grid, and leverage adversarial feature learning to "hallucinate" plausible completions for occluded image parts. We extend several state-of-the-art approaches for road-layout estimation and vehicle occupancy estimation in bird's eye view to the amodal setup and thoroughly evaluate against them. By leveraging temporal sensor fusion to generate training labels, we significantly outperform current art over a number of datasets.*

*Corresponding author: kaustubh3095@gmail.com
Project page: https://hbutsuak95.github.io/monolayout/

1. Introduction

The advent of autonomous driving platforms has led to several interesting, new avenues in perception and scene understanding. While most of the industrially-led solutions leverage powerful sensors (eg. lidar, precision GPS, etc.), an interesting research question is to push the capabilities of monocular vision sensors. To this end, we consider the novel and highly challenging task of estimating *scene layout* in bird's eye view, given only a single color image.

Humans have a remarkable cognitive capability of perceiving *amodal* attributes of objects in an image. For example, upon looking at an image of a vehicle, humans can nominally *infer* the occluded parts, and also the potential geometry of the surroundings of the vehicle. While modern neural networks outperform humans in image recognition and object detection [6,9,12,13,22,24,31], they still lack this innate cognitive capability of reasoning beyond image evidence. With this motivation, we propose *MonoLayout*, a neural architecture that takes as input a color image of a road scene, and outputs the *amodal* scene layout in bird's eye view. *MonoLayout* maps road regions, sidewalks, as well as regions occupied by other traffic participants such as cars, to bird's eye view, in a single pass, leveraging adversarial feature learning.

To the best of our knowledge, *MonoLayout* is the first approach to *amodally* reason about the static and dynamic objects in a scene. We show that, by using a shared context to reason about scene entities, *Mono-*

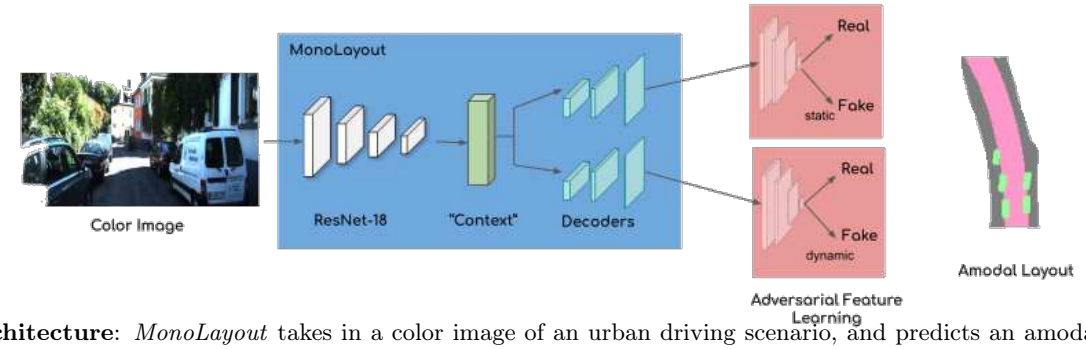

Figure 2: **Architecture**: *MonoLayout* takes in a color image of an urban driving scenario, and predicts an amodal scene layout in bird's eye view. The architecture comprises a *context encoder*, *amodal layout decoders*, and *two discriminators*.

Layout achieves better performance on each task when compared to approaches that train only for a particular task. On the task of amodal scene layout estimation, *MonoLayout* outperforms all evaluated baselines by a significant margin on several subsets of the KITTI [8] and the Argoverse [3] datasets. Further, *MonoLayout* achieves state-of-the-art object detection performance in bird's eye view, without using any form of thresholding / postprocessing. In summary, our contributions are the following:

1. We propose *MonoLayout*, a practically motivated deep architecture to estimate the *amodal* scene layout from just a single image (*c.f*. Fig. 1).
2. We demonstrate that adversarial learning can be used to further enhance the quality of the estimated layouts, specifically when hallucinating large missing chunks of a scene (*c.f*. Fig. 1, Sec. 3).
3. We evaluate against several state-of-the-art approaches, and outperform all of them by a significant margin on a number of established benchmarks (KITTI-Raw, KITTI-Object, KITTI-Odometry [8], Argoverse [3], *c.f*. Sec. 4, Table 1).
4. Further, we show that *MonoLayout* can also be efficiently trained on datasets that do not contain lidar scans by leveraging recent successes in monocular depth estimation. [10] (*c.f*. Table 2).

Please refer to the supplementary material for more results, where we demonstrate that the extracted amodal layouts can suit several higher level tasks, such as (but not limited to) multi-object tracking, trajectory forecasting, etc.

2. Related Work

To the best of our knowledge, no published approach has tackled the task of simultaneous road layout (static scene) and traffic participant (dynamic scene) estimation from a single image. However, several recent approaches have addressed the problem of estimating the layout of a road scene, and several other independent approaches have tackled 3D object detection. We summarize the most closely related approaches in this section.

Road layout estimation

Schulter et al. [26] proposed one of the first approaches to estimate an occlusion-reasoned bird's eye view road layout from a single color image. They use monocular depth estimation [10] as well as semantic segmentation to bootstrap a CNN that predicts occluded road layout. They use priors from OpenStreetMap [21] to adversarially regularize the estimates. More recently, Wang *et al*. [29] builds on top of [26] to infer parameterized road layouts. Our approach does not require to be bootstrapped using either semantic segmentation or monocular depth estimation, and can be trained end-to-end from color images.

Perhaps the closest approach to ours is MonoOccupancy [19], which builds a variational autoencoder (VAE) to predict road layout from a given image. They also present results for extracting regions close to roads (eg. sidewalk, terrain, non-free space). However, they reason only about the pixels present in the image, and not beyond occluding obstacles. Also, the bottleneck enforced by that leads to non-sharp, blob-like layouts. On the other hand, *MonoLayout* estimates *amodal* scene layouts, reasoning beyond occlusion boundaries. Our approach produces crisp road edges as well as vehicle boundaries, by leveraging adversarial feature learning and sensor fusion to reduce noise in the labeled ground-truth training data.

Object detection in bird's eye view

There exist several approaches to 3D object detection that exclusively use lidar [2,27,30], or a combination of camera and lidar sensors [5,15,17]. However, there are only a handful of approaches that purely use monocular vision for object detection [4, 16, 20]. Most of these are two stage approaches, comprising a region-proposal stage, and a classification stage.

Another category of approaches map a monocular image to a bird's eye view representation [23], thereby reducing the task of 3D object detection to that of 2D image segmentation. Recently, BirdGAN [28] leveraged adversarial learning for mapping images to bird's

eye view, where lidar object detectors such as [2] were repurposed for object detection.

Such techniques usually require a pre-processing stage (usually a neural network that maps an image to a bird's eye view) after which further processing is applied. On the other hand, we demonstrate that we can achieve significantly higher accuracy by directly mapping from the image space to objects in bird's eye view, bypassing the need for a pre-processing stage altogether.

More notably, all the above approaches require a post-processing step that usually involves non-maximum suppression / thresholding to output object detections. *MonoLayout* neither requires pre-processing nor post-processing and it directly estimates scene layouts that can be evaluated (or plugged into other task pipelines).

3. MonoLayout: Monocular Layout Estimation

3.1. Problem Formulation

In this paper, we address the problem of *amodal* scene layout estimation from a single color image. Formally, given a color image $\mathcal{I}$ captured from an autonomous driving platform, we aim to predict a bird's eye view layout of the static and dynamic elements of the scene. Concretely, we wish to estimate the following three quantities. [1]

1. The set of all static scene points $\mathcal{S}$ (typically the road and the sidewalk) on the ground plane (within a rectangular range of length L and width W, in front of the camera), *regardless of whether or not they are imaged by the camera*.
2. The set of all dynamic scene points $\mathcal{D}$ on the ground plane (within the same rectangular range as above) occupied by vehicles, *regardless of whether or not they are imaged by the camera*.
3. For each point discerned in the above step as being occupied by a vehicle, an instance-specific labeling of which vehicle the point belongs to.

3.2. MonoLayout

The problem of amodal scene layout estimation is challenging from a neural networks standpoint in several interesting ways. First, it necessitates that we learn *good* visual representations from images that help in estimating 3D properties of a scene. Second, it requires these representations to reason beyond classic 3D reconstruction; these representations must enable us to *hallucinate* 3D geometries of image regions that are occupied by occluders. Furthermore, the learned representations must implicitly disentangle the static parts of the scene (occupied by road points) from the dynamic objects (eg. parked/moving cars). With these requirements in mind, we design *MonoLayout* with the following components.

Maximum a posteriori estimation

We formulate the amodal road layout estimation problem as that of recovering the Maximum a posteriori (MAP) estimate of the distribution of scene statics and dynamics. Given the image $\mathcal{I}$, we wish to recover the posterior $P(\mathcal{S}, \mathcal{D}|\mathcal{I})$, over the domain $\Omega \triangleq \{(x, H, z) | \|(x - x_0)\|_1 \leq L;\ \ \|(z - z_0)\|_1 \leq W; (z - z_0) > 0\}$[2]. Note that the static (road) and dynamic (vehicle) marginals are not independent. They are not independent - they exhibit high correlation (vehicles ply on roads). Hence, we introduce an additional conditioning context variable $\mathcal{C}$ that can be purely derived only using the image information I, such that, $\mathcal{S}$ and $\mathcal{D}$ are conditionally independent given $\mathcal{C}$. We term this conditioning variable as the "shared context" as it necessarily encompasses the information needed to estimate the static and dynamic layout marginals. This allows the posterior to be factorized in the following form.

$$\begin{aligned} P(\mathcal{S}, \mathcal{D}|\mathcal{I}) &\propto P(\mathcal{S}, \mathcal{D}, \mathcal{C}|\mathcal{I}) \\ &= \underbrace{P(\mathcal{S}|\mathcal{C}, \mathcal{I})}_{\text{static decoder}} \quad \underbrace{P(\mathcal{D}|\mathcal{C}, \mathcal{I})}_{\text{dynamic decoder}} \quad \underbrace{P(\mathcal{C}|\mathcal{I})}_{\text{context encoder}} \end{aligned} \tag{1}$$

In accordance with the above factorization of the posterior, the architecture of *MonoLayout* comprises three subnetworks (*c.f.* Fig. 2).

1. A *context encoder* which extracts multi-scale feature representations from the input monocular image. This provides a shared context that captures static as well as dynamic scene components for subsequent processing.
2. An *amodal static scene decoder* which decodes the shared context to produce an amodal layout of the static scene. This model consists of a series of deconvolution and upsampling layers that map the shared context to a static scene bird's eye view.
3. A *dynamic scene decoder* which is architecturally similar to the road decoder and predicts the vehicle occupancies in bird's eye view.

[1] *Flat-earth assumption*: For the scope of this paper, we assume that the autonomous vehicle is operating within a bounded geographic area of the size of a typical city, and that all roads in consideration are *somewhat* planar, i.e., no steep/graded roads on mountains.

[2] This domain is a rectangular region in bird's eye view. H is the height of the camera above the ground.

4. Two *discriminators* [14, 25] which regularize the predicted static/dynamic layouts by regularizing their distributions to be *similar* to the *true* distribution of plausible road geometries (which can be easily extracted from huge unpaired databases such as OpenStreetMap [21]) and ground-truth vehicle occupancies.

Feature Extractor

From the input image, we first extract meaningful image features at multiple scales, using a ResNet-18 encoder (pre-trained on ImageNet [7]). We finetune this feature extractor in order for it to learn low-level features that help reason about static and dynamic aspects of the scene.

Static and dynamic layout decoders

The static and dynamic layout decoders share an identical architecture. They decode the shared context from the feature extractor by a series of upsampling layers to output a $D \times D$ grid each[3].

Adversarial Feature Learning

To better ground the likelihoods $P(\mathcal{S}|\mathcal{C},\mathcal{I})$, $P(\mathcal{D}|\mathcal{C},\mathcal{I})$ ($c.f.$ Eq. 1), we introduce adversarial regularizers (discriminators) parameterized by θ_S and θ_D respectively. The layouts estimated by the static and dynamic decoders are input to these patch-based discriminators [14]. The discriminators regularize the distribution of the output layouts (*fake data* distribution, in GAN [11] parlance) to match a prior data distribution of *conceivable* scene layouts (*true data* distribution). This prior data distribution is a collection of road snippets from OpenStreetMap [21], and rasterized images of vehicles in bird's eye view. Instead of training with a paired OSM for each image, we choose to collect a set of diverse OSM maps representing the true data distribution of road layouts in bird's eye view and train our discriminators in an unpaired fashion. This mitigates the need to have perfectly aligned OSM views to the current image, making *MonoLayout* favourable compared to approaches like [26] that perform an explicit alignment of the OSM before beginning processing.

Loss function

The parameters ϕ, ν, ψ of the context encoder, the amodal static scene decoder, and the dynamic scene decoder respectively are obtained by minimizing the following objective using minibatch stochastic gradient descent.

$$\min_{\phi,\nu,\psi,\theta_S,\theta_D} \mathcal{L}_{sup}(S,D;\phi,\nu,\psi) + \mathcal{L}_{adv}(S,D;\phi,\theta,\psi) + \mathcal{L}_{discr}(D_S,D_D;\phi,\nu)$$

$$\mathcal{L}_{sup} = \sum_{i=1}^{N} \|\mathcal{S}_{\phi,\nu}(\mathcal{I}^i) - \mathcal{S}^i_{gt}\|^2 + \|\mathcal{D}_{\phi,\psi}(\mathcal{I}^i) - \mathcal{D}^i_{gt}\|^2 \tag{2}$$

$$\mathcal{L}_{adv}(S,D;\phi,\theta,\psi) = \mathbb{E}_{\theta\sim p_{fake}}\left[(D(\theta_S)-1)^2\right] + \mathbb{E}_{\theta\sim p_{fake}}\left[(D(\theta_D)-1)^2\right]$$

$$\mathcal{L}_{discr}(D;\theta) = \sum_{\theta\in\{\theta_D,\theta_S\}} \mathbb{E}_{\theta\sim p_{true}}\left[(D(\theta)-1)^2\right] + \mathbb{E}_{\theta\sim p_{fake}}\left[(D(\theta)-1)^2\right]$$

Here, $\mathcal{L}_{sup}$ is a supervised ($L2$) error term that penalizes the deviation of the predicted static and dynamic layouts ($\mathcal{S}_{\phi,\nu}(\mathcal{I})$), $\mathcal{D}_{\phi,\nu}(\mathcal{I})$)) with their corresponding ground-truth values ($\mathcal{S}^i_{gt}$, $\mathcal{D}^i_{gt}$). The adversarial loss $\mathcal{L}_{adv}$ encourages the distribution of layout estimates from the static/dynamic scene decoders (p_{fake}) to be close to their true counterparts (p_{true}). The discriminator loss $\mathcal{L}_{discr}$ is the discriminator update objective [11].

3.3. Generating training data: sensor fusion

Since we aim to recover the amodal scene layout, we are faced with the problem of extracting training labels for even those parts of the scene that are occluded from view. While recent autonomous driving benchmarks provide synchronized lidar scans as well as semantic information for each point in the scan, we propose a sensor fusion approach to generate more robust training labels, as well as to handle scenarios in which direct 3D information (eg. lidar) may not be available.

As such, we use either monocular depth estimation networks (Monodepth2 [10]) or raw lidar data and initialize a pointcloud in the camera coordinate frame. Using odometry information over a window of W frames, we aggregate/register the sensor observations over time, to generate a more dense, noise-free pointcloud. Note that, when using monocular depth estimation, we discard depths of points that are more than 5 meters away from the car, as they are noisy. To compensate for this narrow field of view, we aggregate depth values over a much larger window size (usually $40-50$) frames.

The dense pointcloud is then projected to an occupancy grid in bird's eye view. If ground-truth semantic segmentation labels are available, each occupancy grid cell is assigned the label based on a simple majority over the labels of the corresponding points. For the case where ground-truth semantic labels are unavailable, we use a state-of-the-art semantic segmentation network [24] to segment each frame and aggregate these predictions into the occupancy grid.

For vehicle occupancy estimation though, we rely on ground-truth labels in bird's eye view, and train only

[3]We tried training a single decoder for both the tasks, but found convergence to be hard. We attribute this to the extreme change in output spaces: while roads are large, continuous chunks of space, cars are tiny, sparse chunks spread over the entire gamut of pixels. Instead, we chose to train two decoders over a shared context, which bypasses this discrepancy in output spaces, and results in sharp layout estimates.

Dataset	Method	Static Layout Estimation			Vehicle Layout	
		Road	Sidewalk	Road + Sidewalk		
		mIoU	mIoU	occl mIoU	mIoU	mAP
KITTI Raw	MonoOccupancy [19]	56.16	18.18	28.24	-	-
	Schulter *et al.* [26]	68.89	30.35	61.06	-	-
	MonoLayout-static (Ours)	**73.86**	**32.86**	**67.42**	-	-
KITTI Odometry	MonoOccupancy [19]	64.72	12.08	34.87	-	-
	MonoLayout-static (Ours)	**80.08**	**42.66**	**72.46**	-	-
KITTI Object	MonoOccupancy-ext	-	-	-	20.45	22.59
	Mono3D [4]	-	-	-	17.11	29.62
	OFT [23]	-	-	-	25.24	34.69
	MonoLayout-dynamic (Ours)	-	-	-	**26.08**	**40.79**
KITTI Tracking	**MonoLayout (Ours)**	**53.94**	-	-	**24.16**	**36.83**
Argoverse	MonoOccupancy-ext	32.70	-	-	16.22	38.66
	MonoLayout (Ours)	**58.33**	-	-	**32.05**	**48.31**

Table 1: **Quantitative results**: We evaluate the performance of *MonoLayout* on several datasets, on amodal scene layout estimation. As there is no existing baseline that simultaneously estimates static (road/sidewalk) as well as dynamic (vehicle) layout, we evaluate under multiple settings. On the KITTI Raw and KITTI Odometry [8] datasets, we evaluate *MonoLayout*-static. On the KITTI Object [8] dataset, we evaluate *MonoLayout*-dynamic. On the KITTI Tracking [8] and Argoverse [3] datasets, we evaluate *MonoLayout*, the full architecture that estimates both static and dynamic layouts. We outperform existing approaches by a significant margin, on all metrics.

on datasets that contain such labels[4].

4. Experiments

To evaluate *MonoLayout*, we conduct experiments over a variety of challenging scenarios and against multiple baselines, for the task of *amodal* scene layout estimation.

4.1. Datasets

We present our results on two different datasets - KITTI [8] and Argoverse [3]. The latter contains a *high-resolution* semantic occupancy grid in bird's eye view, which facilitates the evaluation of amodal scene layout estimation. The KITTI dataset, however, has no such provision. For a semblance of *ground-truth*, we register depth and semantic segmentation of lidar scans in bird's eye view.

To the best of our knowledge, there is no published prior work that reasons jointly about road and vehicle occupancies. However, there exist approaches for road layout estimation [19, 26], and a separate set of approaches for vehicle detection [4, 23]. Furthermore, each of these approaches evaluate over different datasets (and in cases, different train/validation splits). To ensure fair comparision with all such approaches, we organize our results into the following categories.

1. **Baseline Comparison:** For a fair comparison with state-of-the-art road layout estimation techniques, we evaluate performance on the KITTI RAW split used in [26] (10156 training images, 5074 validation images). For a fair comparision with state-of-the-art 3D vehicle detection approaches we evaluate performance on the KITTI 3D object detection split of Chen et al. [4] (3712 training images, 3769 validation images).
2. **Amodal Layout Estimation:** To evaluate layout estimation on both static and dynamic scene attributes (road, vehicles), we use the KITTI Tracking [8] and Argoverse [3] datasets. We annotate sequences from the KITTI Tracking dataset for evaluation (5773 training images, 2235 validation images). Argoverse provides HD maps as well as vehicle detections in bird's eye view (6723 training images, 2418 validation images).
3. **Temporal sensor fusion for supervision:** We then present results using our data generation approach (*c.f.* Sec. 3.3) on the KITTI Odometry dataset. This also uses the dense semantic segmentation labels from the Semantic KITTI dataset [1].

[4]For a detailed description of the architecture and the training process, we refer the reader to the supplementary material

4.2. Approaches evaluated

We evaluate the performance of the following approaches.

- *Schulter* et al.: The static scene layout estimation approach proposed in [26].
- *MonoOccupancy*: The static scene layout estimation approach proposed in [19].
- *Mono3D*: The monocular 3D object detection approach from [4].
- *OFT*: A recent, state-of-the-art monocular bird's eye view detector [23].
- *MonoOccupancy-ext*: We extend MonoOccupancy [19] to predict vehicle occupancies.
- *MonoLayout-static*: A version of *MonoLayout* that only predicts static scene layouts.

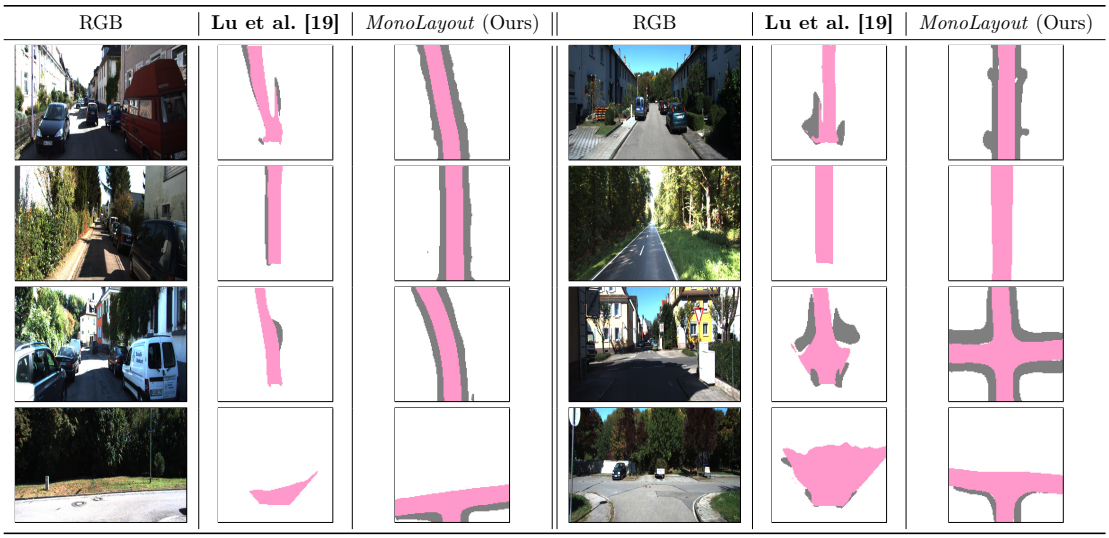

Figure 3: **Static layout estimation**: Observe how *MonoLayout* performs amodal completion of the static scene (road shown in pink, sidewalk shown in gray. MonoOccupancy [19] fails to reason beyond occluding objects (top row), and does not hallucinate large missing patches (bottom row), while *MonoLayout*(Ours) is accurately able to do so. Furthermore, even in cases where there is no occlusion (row 2), *MonoLayout*(Ours) generates road layouts of much sharper quality. Row 3 show extremely challenging scenarios where most of the view is blocked by vehicles, and the scenes exhibit high-dynamic range (HDR) and shadows.

- *MonoLayout-dynamic*: A version of *MonoLayout* that only predicts vehicle occupancies.
- *MonoLayout*: The full architecture, that predicts both static and dynamic scene layouts.

4.3. Static layout estimation (Road)

We evaluate *MonoLayout*-static against Schulter *et al*. [26] and MonoOccupancy [19] on the task of static scene (road/sidewalk) layout estimation. Note that Schulter *et al*. assume that the input image is first passed through monocular depth estimation and semantic segmentation networks, while we operate directly on raw pixel intensities. Table 1 summarizes the performance of existing road layout estimation approaches (Schulter *et al*. [26], MonoOccupancy [19]) on the KITTI Raw and KITTI Odometry benchmarks. For KITTI Raw, we follow the exact split used in Schulter *et al*. and retrain MonoOccupancy and *MonoLayout*-static on this train split. Since the manual annotations for semantic classes for KITTI Raw aren't publicly available, we manually annotated sequences with semantic labels (and will make them publicly available).

From Table 1 (*c.f*. "KITTI Raw"), we see that *MonoLayout*-static outperforms both MonoOccupancy [19] and Schulter *et al*. [26] by a significant margin. We attribute this to the strong hallucination capabilities of *MonoLayout*-static due to adversarial feature learning (*c.f*. Fig. 3). Although Schulter *et al*. [26] use a discriminator to regularize layout predictions, they seem to suffer from cascading errors due to sequential processing blocks (eg. depth, semantics extraction). MonoOccupancy [19] does not output sharp estimates of road boundaries by virtue of being a variational autoencoder (VAE), as mean-squared error objectives and Gaussian prior assumptions often result in *blurry* generation of samples [18]. The hallucination capability is much more evident in the occluded region evaluation, where we see that *MonoLayout*-static improves by more than 10% on prior art.

4.4. Dynamic (vehicle) layout estimation

To evaluate vehicle layout estimation in bird's eye view, we first present a comparative analysis on the KITTI Object [8] dataset, for a fair evaluation with respect to prior art. Specifically, we compare against *Orthographic Feature Transform* (OFT [23]), the current best monocular object detector in bird's eye view. As a baseline approach, we also evaluate against Mono3D [4]. Furthermore, we extend MonoOccupancy [19] to the task of vehicle layout estimation, to demonstrate that variational autoencoder-style architectures are ill-suited to this purpose. This comparison is presented in Table 1 ("KITTI Object").

We see again that *MonoLayout*-dynamic outperforms prior art on the task of vehicle layout estimation. Note that Mono3D [4] is a two-stage method and requires strictly additional information (semantic and instance segmentation), and OFT [23] performs explicit orthographic transfors and is parameter-heavy (23.5M parameters) which slows it down considerably (5 fps). We make no such assumptions and operate on raw image intensities, yet obtain better performance, and about a 6x speedup (32 fps). Also, MonoOccupany [19] does not perform well on the vehicle occupancy estimation task, as the variational autoencoder-style architecture usually *merges* most of the vehicles into large *blob-like* structures (*c.f*. Fig. 4).

4.5. Amodal scene layout estimation

In the previous sections, we presented results individually for the tasks of static (road) and dynamic (vehicle) layout estimation, to facilitate comparision with prior art *on equal footing*. We now present results for *amodal* scene

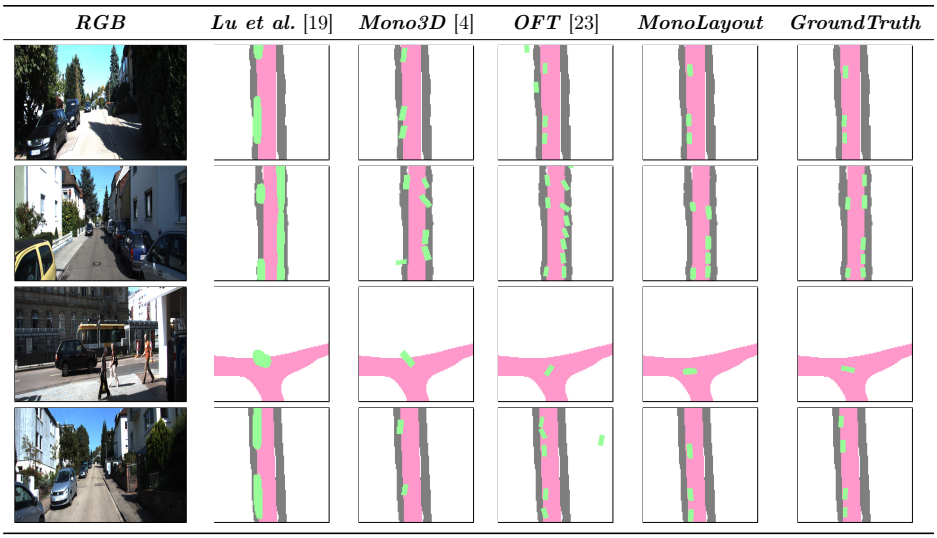

Figure 4: **Dynamic layout estimation**: We show vehicle occupancy estimation results on the KITTI [8] 3D Object detection benchmark. From left to right, the column corresponds to the input image, MonoOccupancy [19], Mono3D [4], OFT [23], *MonoLayout* (Ours), and ground-truth respectively. While the other approaches miss out on detecting cars (top row), or split a vehicle detection into two (second row), or stray detections off road (third row), *MonoLayout* (Ours) produces crisp object boundaries while respecting vehicle and road geometries.

layout estimation (i.e., both static and dynamic scene components) on the Argoverse [3] dataset. We chose Argoverse [3] as it readily provides ground truth HD bird's eye view maps for both road and vehicle occupancies. We follow the train-validation splits provided by [3], and summarize our results in Table 1 ("Argoverse").

We show substantial improvements at nearly 20% on mIoU vis-a-vis the next closest baseline [19]. This is a demonstration of the fact that, even when perfect ground-truth is available, approaches such as MonoOccupancy [19] fail to reach the levels of performance as that of *MonoLayout*. We attribute this to the *shared context* that encapsulates a rich feature collection to facilitate both static and dynamic layout estimation. Note that, for the Argoverse [3] dataset we *do not* train our methods on the ground-truth HD maps, because such fine maps aren't typically available for all autonomous driving solutions. Instead, we train our methods using a semblance of ground-truth (generated by the process described in Sec 3.3), and use the HD maps only for evaluation. This validates our claim that our model, despite being trained using noisy ground estimates by leveraging sensor fusion, is still able to hallucinate and complete the occluded parts of scenes correctly as shown in Fig 6.

4.6. Ablation Studies

Using monocular depth as opposed to lidar

While most of the earlier results focused on the scenario where explicit lidar annotations were available, we turn to the more interesting case where the dataset only comprises monocular images. As described in Sec 3.3, we use monocular depth estimation (MonoDepth2 [10]) and aggregate/register depthmaps over time, to provide training signal. In Table 2, we analyze the impact of the availability

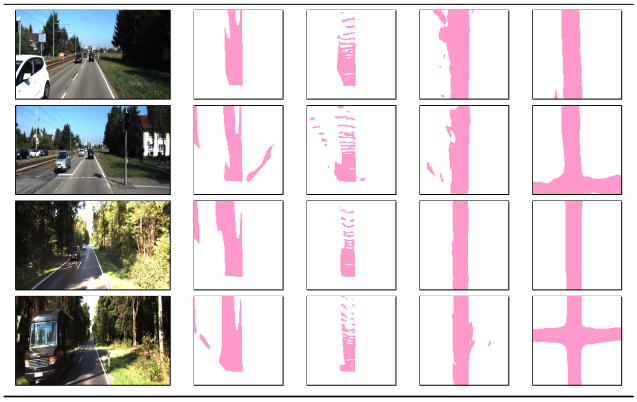

Figure 5: **Impact of sensor fusion**: *Col 1*: Input images. *Col 2*: Using per-frame monocular depth, *Col 3*: Using per-frame lidar depth, *Col 4*: Sensor-fused monocular depth, *Col 5*: Sensor-fused lidar depth. (Refer to Table 2 and Sec. 4.6 for more details)

of lidar data on the performance of amodal scene layout estimation.

We train MonoOccupancy [19] and *MonoLayout*-static on the KITTI Raw dataset, using monocular depth estimation-based ground-truth, as well as lidar-based ground-truth. While lidar based variants perform better (as is to be expected), we see that self-supervised monocular depth estimation results in reasonable performance too. Surprisingly, for the per-frame case (i.e., no sensor fusion), monocular depth based supervision seems to fare better. Under similar conditions of supervision, we find that *MonoLayout*-static outperforms MonoOccupancy [19].

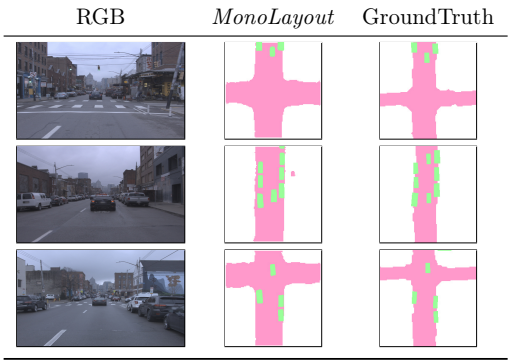

Figure 6: **Amodal scene layout estimation** on the Argoverse [3] dataset. The dataset comprises multiple challenging scenarios, with low illumination, large number of vehicles. *MonoLayout* is accurately able to produce sharp estimates of vehicles and road layouts. (Sidewalks are not predicted here, as they aren't annotated in Argoverse).

Impact of sensor fusion

The sensor fusion technique described in Sec 3.3 greatly enhances the accuracy of static layout estimation. Aggregating sensor observations over time equips us with more comprehensive, and noise-free maps. Table 2 presents results for an analysis of the performance benefits obtained due to temporal sensor fusion.

Supervision	*MonoOccupancy-ext*	*MonoLayout*-static (Ours)
Per-frame monocular depth	56.16	58.87
Sensor-fused monocular depth	64.81	66.71
Per-frame lidar	44.29	48.29
Sensor-fused lidar	**71.67**	**73.86**

Table 2: **Monocular depth**: If lidar data is unavailable, we leverage self-supervised monocular depth estimation to generate training data for *MonoLayout*-static and achieve reasonable static layout estimation (rows 1-2). Although performance is inferior to the case when lidar is available (rows 3-4), this is not unexpected. **Sensor fusion**: Regardless of the modality of depth information, sensor-fusing depth estimates over a window of 40 frames dramatically improves performance (row 2, row 4).

Impact of adversarial learning

With the discriminators, we not only improve qualitatively (sharper/realistic samples) (*c.f.* Fig 7), but also gain significant performance boosts. This is even more pronounced on Argoverse [3], as shown in Table 3. In case of vehicle occupancy estimation, while using a discriminator does not translate to quantitative performance gains, it often results in qualitatively sharper, aesthetic estimates as seen in Fig 7.

Timing analysis

We also show the computation test time of our method as compared to other baslines in Table 4. Unlike Schulter *et al.* [26], our network does not require discriminator to be used during inference time. It achieves real time inference rate of approx. 32 Hz for an input size $3 \times 512 \times 512$ and an output size $2 \times 128 \times 128$ on an NVIDIA GeForce GTX 1080Ti GPU. Note that in *MonoLayout* the static and dynamic decoders are executed in parallel, maintining comparable runtime. *MonoLayout* is an order of magnitude faster than previous methods, making it more attractive for on-road implementations.

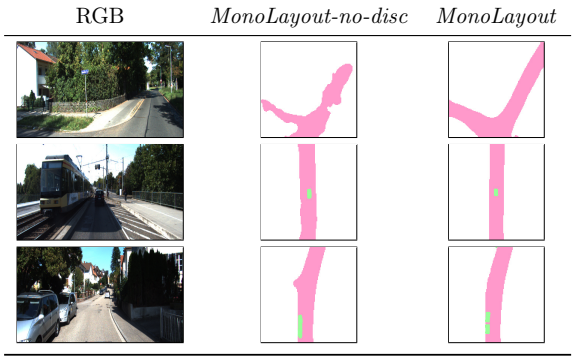

Figure 7: **Effect of adversarial learning**: As can be clearly seen here, the discriminators help enhance both the static (road) layout estimation (top and middle rows), as well as produce sharper vehicle boundaries (bottom row). While this translates to performance gains in static layout estimation (*c.f.* Table 3), the gains in dynamic layout estimation are more cosmetic in nature.

Dataset	***MonoLayout-no-disc***			***MonoLayout***		
	Road	**Vehicle (mIoU)**	**Vehicle (mAP)**	**Road**	**Vehicle (mIoU)**	**Vehicle (mAP)**
KITTI Raw	70.95	-	-	**73.86**	-	-
KITTI Object	-	**26.25**	37.66	-	25.47	**41.52**
Argoverse	51.66	**32.84**	44.07	**58.33**	32.06	**48.31**

Table 3: **Effect of discriminator**: Adding a discriminator clearly translates to an accuracy boost in static (road) layout estimation. For vehicle occupancy estimation, while a quantitative boost is not perceived, the generated layouts are sharper, and aesthetic as opposed to when not using the discriminator (*c.f.* Fig. 7)

Method	**Parameters**	**Computation Time**
Mono3D [4]	>> 20 M	0.24 fps
OFT [23]	23.5 M	< 5 fps
MonoOccupancy [19]	27.5 M	15 fps
Schulter et al. [26]	>> 20 M	< 3 fps
MonoLayout (Ours)	19.6M	32 fps

Table 4: A comparative study of infrence time of various methods. *MonoLayout* is about as faster and significantly more accurate compared to prior art. (*c.f.* Table 1).

5. Discussion and conclusions

This paper proposed *MonoLayout*, a versatile deep network architecture capable of estimating the amodal layout of a complex urban driving scene in real-time. In the supplementary material, we show several additional results, including extended ablations, and applications to multi-object tracking and trajectory forecasting. A promising avenue for future research is the generalization of *MonoLayout* to unseen scenarios, as well as incorporating temporal information to improve performance.

References

[1] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *ICCV*, 2019.

[2] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. Garcia, and A. De La Escalera. Birdnet: a 3d object detection framework from lidar information. In *ITSC*, 2018.

[3] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *CVPR*, 2019.

[4] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3d object detection for autonomous driving. In *CVPR*, 2016.

[5] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017.

[6] D. Cireşan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. *arXiv preprint*, 2012.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[8] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[9] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[10] C. Godard, O. Mac Aodha, M. Firman, and G. Brostow. Digging into self-supervised monocular depth estimation. *arXiv preprint*, 2018.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*. 2014.

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[14] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[15] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. Joint 3d proposal generation and object detection from view aggregation. In *IROS*, 2018.

[16] B. Li, W. Ouyang, L. Sheng, X. Zeng, and X. Wang. Gs3d: An efficient 3d object detection framework for autonomous driving. In *CVPR*, 2019.

[17] M. Liang, B. Yang, S. Wang, and R. Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *ECCV*, 2018.

[18] S. Lin, R. Clark, R. Birke, N. Trigoni, and S. Roberts. Wise-ale: Wide sample estimator for aggregate latent embedding. 2019.

[19] C. Lu, M. J. G. van de Molengraft, and G. Dubbelman. Monocular semantic occupancy grid mapping with convolutional variational encoder–decoder networks. *IEEE Robotics and Automation Letters*, 2019.

[20] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3d bounding box estimation using deep learning and geometry. In *CVPR*, 2017.

[21] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org , 2017.

[22] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[23] T. Roddick, A. Kendall, and R. Cipolla. Orthographic feature transform for monocular 3d object detection. *arXiv preprint*, 2018.

[24] S. Rota Bulò, L. Porzi, and P. Kontschieder. In-place activated batchnorm for memory-optimized training of dnns. In *CVPR*, 2018.

[25] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, 2016.

[26] S. Schulter, M. Zhai, N. Jacobs, and M. Chandraker. Learning to look around objects for top-view representations of outdoor scenes. In *ECCV*, 2018.

[27] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019.

[28] S. Srivastava, F. Jurie, and G. Sharma. Learning 2d to 3d lifting for object detection in 3d for autonomous vehicles. *arXiv preprint*, 2019.

[29] Z. Wang, B. Liu, S. Schulter, and M. Chandraker. A parametric top-view representation of complex road scenes. In *CVPR*, 2019.

[30] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3d object detection from point clouds. In *CVPR*, 2018.

[31] Q. Yu, Y. Yang, F. Liu, Y.-Z. Song, T. Xiang, and T. M. Hospedales. Sketch-a-net: A deep neural network that beats humans. *International journal of computer vision*, 2017.