

# RackLay: Multi-Layer Layout Estimation for Warehouse Racks

Meher Shashwat Nigam<sup>\*1,2</sup>, Avinash Prabhu<sup>\*1</sup>, Anurag Sahu<sup>\*1</sup>, Puru Gupta<sup>†1</sup>, Tanvi Karandikar<sup>†1</sup>,  
N. Sai Shankar<sup>1</sup>, Ravi Kiran Sarvadevabhatla<sup>2</sup>, and K. Madhava Krishna<sup>1</sup>

<sup>1</sup>Robotics Research Center, IIIT Hyderabad, <sup>2</sup>Center for Visual Information Technology, IIIT Hyderabad

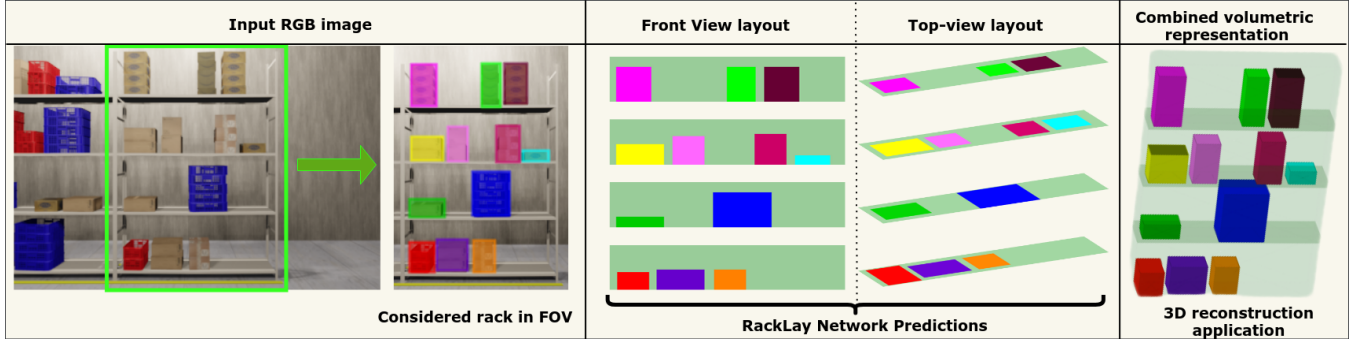


Fig. 1: Given a monocular RGB image of a warehouse rack, we propose **RackLay**, a deep neural architecture that generates the *top-view* and *front-view* semantic layout for rack shelves and items placed on each shelf. Fusing these layouts provides a volumetric reconstruction of the rack, enabling 3D reasoning. For the considered rack in the figure, our system can report "Rack has 4 shelves, 12 box stacks, and 830cm<sup>3</sup> of free space available".

**Abstract**—Given a monocular color image of a warehouse rack, we aim to predict the *bird’s-eye view* layout for each shelf in the rack, which we term as ‘multi-layer’ layout prediction. To this end, we present *RackLay*, a deep neural network for real-time shelf layout estimation from a single image. Unlike previous layout estimation methods which provide a single layout for the dominant ground plane alone, *RackLay* estimates the top-view and front-view layout for each shelf in the considered rack populated with objects. *RackLay*’s architecture and its variants are versatile and estimate accurate layouts for diverse scenes characterized by varying number of visible shelves in an image, large range in shelf occupancy factor and varied background clutter. Given the extreme paucity of datasets in this space and the difficulty involved in acquiring real data from warehouses, we additionally release a flexible synthetic dataset generation pipeline *WareSynth* which allows users to control the generation process and tailor the dataset according to contingent application. The ablations across architectural variants and comparison with strong prior baselines vindicate the efficacy of *RackLay* as an apt architecture for the novel problem of multi-layered layout estimation. We also show that fusing the top-view and front-view enables 3D reasoning applications such as metric free space estimation for the considered rack.

## I. INTRODUCTION

The importance and the necessity of warehouse automation grows by the day and the future is painted with a scenario where a fleet of robots manage the warehouse with or without human intervention [1]. Nonetheless, almost 30% of the warehouses today operate without their staple warehouse management systems (WMS)[2]. In such situations, essential tasks such as shelf occupancy estimation and management become important and challenging tasks.

In this paper, we address the hitherto untackled problem of layout and freespace estimation for rack shelves. This problem is equally important in the context of warehouses without WMS as well as in the scenarios where automated robotic agents manipulate a shelf space. In this effort, monocular vision is the sensing modality considering the ubiquitous, low cost, high portability and scalability of such single camera systems.

We propose a simple yet effective network architecture *RackLay*<sup>1</sup>, which takes a single RGB image as input and outputs the top-view and front-view layouts of all the shelves comprising the dominant rack in the image (see Fig. 1). *RackLay* consists of a shared context encoder which reasons about the shelves and objects together. In turn, such reasoning enables *RackLay*’s decoder to generate both the *top-view* and *front-view* layout of the rack on a per-shelf basis (see Fig. 2).

It is important to note that the problem is not immediately reducible to any standard formulation of object recognition, layout estimation or semantic segmentation. Objects on the rack shelves are amenable for semantic segmentation [3] or object detection [4]. However, this is not the case for racks themselves, which appear as occluded, diffused, thin structures. Indeed, these hollow structures pose a challenge for mainstream approaches. For very similar reasons, existing approaches cannot be trivially adapted for localizing rack shelves. Unlike standard layout formulations which estimate the layout with reference to a single dominant plane (e.g. ground plane) [5], warehouse rack shelves are disconnected and distinct planar segments present at multiple heights (lay-

<sup>\*</sup>,<sup>†</sup> denote equal contribution

Corresponding author: meherashashwat@gmail.com

<sup>1</sup>Project page: <https://github.com/Avinash2468/RackLay>

ers) relative to ground plane. Needless to say, these shelves can be either empty or contain an arbitrary number of objects. Hence, a cornerstone novelty of the present formulation is the adaptation of deep architectures to the problem of layout estimation over multiple shelf levels (layers) that constitute a rack and contents thereof.

Specifically the paper contributes as follows:

- 1) It solves for the first time, the problem of shelf layout estimation for warehouse rack scenes – a problem pertinent in the context of both warehouse inventory management as well as futuristic warehouses managed by an autonomous robotic fleet.
- 2) It proposes a novel architecture, (Sec. IV), the keynote of which is a shared context encoder, and most importantly a multi-channel decoder that infers the layout for each and every shelf in a given rack. We release for the first time, the *RackLay* synthetic dataset consisting of 20k RGB images along with layout annotations of shelves and objects in both the top and front view.
- 3) More importantly, we open-source the flexible data generation pipeline *WareSynth*, along with relevant instructions that enable the researcher/user to create and customize their own warehouse scenes and generate 2D/3D ground truth annotations needed for their task automatically<sup>2</sup>. This does not restrict or limit the user to our dataset alone but provides for possibilities to create new datasets with the ability to customize as desired, as discussed in detail in Sec. III.
- 4) We show tangible performance gain compared to other baseline architectures [6] dovetailed and adapted to the problem of rack layout estimation. Moreover, we tabulate a number of ablations across architectural variants which establish the efficacy and superiority of *RackLay* (Sec. V).

## II. RELATED WORK

In recent years, learning scene layouts and obtaining volumetric representations directly from an RGB image has garnered lot of interest. Deep learning methods have become more reliable and accurate for many computer vision tasks like object detection, semantic segmentation and depth estimation. But even a combination of these fundamental solutions does not suffice for higher-order tasks like shelf-layout estimation in warehouse management systems, which requires multi-layer top-view layout estimation. To that extent, we summarize the existing approaches and differentiate our method from the rest.

1) *Indoor scene understanding*: Room layout estimation from a single image has been the most solved problem [7], [8] in the context of indoor 3D scene understanding. There have also been a few approaches for amodal perception as well-[9], [10]. Indoor scene understanding can rely on strong assumptions like a Manhattan world layout, which works well for single room layouts.

2) *Object detection methods*: We relate to deep learning based detection models, as a large part of our problem deals with localizing semantic classes like shelves and boxes/cartons in an 3D scene. Several existing approaches aim to detect object layouts in 3D. Some of these [11], [12] approaches combine information from images and LiDAR, others [5], [13] work by converting images to *bird’s eye view* representations, followed by object detection.

3) *Bird’s eye view (BEV) representation*: BEV semantic segmentation has been tackled mostly for outdoor scenes. Gupta *et al.*[14] demonstrate the suitability of a BEV representation for mapping and planning. Schuler *et al.*[15] 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 and semantic segmentation to aid their network that predicts occluded road layout. Wang *et al.*[16] builds on top of [15] to infer parameterized road layouts. Parametric models might not account for all possible scene layouts, whereas our approach is non-parametric and thus more flexible. MonoOccupancy [17], uses a variational autoencoder (VAE) to predict road layout from a given image, but only for the pixels present in the image. MonoLayout [18], can be trained end to end on colour images, reasons beyond occlusion boundaries and does not need to be bootstrapped with these additional inputs. We predict the occlusion-reasoned occupancy layouts of multiple parallel planes(layers), with varying height, from a single view RGB image.

4) *Warehouse Datasets*: There are very few datasets publicly available for warehouse settings. Real-world datasets like LOCO[19] exist for scene understanding in warehouse logistics, in which they provide a limited number of RGB images, along with corresponding 2D annotations. Due to the difficulty in acquiring the 3D ground truth information from real scenes there aren’t any real warehouse datasets which provide information about the objects in scene and their relative positions and orientation. For 3D deep learning applications, large amount of diverse data along with 3D ground truth information is required. There are general purpose synthetic data simulators like NVIDIA Isaac [20], which provide warehouse scenes. However, they provide lesser control as to specifying properties for the warehouse, and can’t be modified easily to generate annotations needed for our task.

## III. DATASET GENERATION PIPELINE

In this section, we introduce our synthetic data generation pipeline termed *WareSynth*, which can be used to generate 3D warehouse scenes, automate the process of data capture and generate corresponding annotation.

### A. Software and initial models

For modelling the warehouse setup, we used the open source 3D graphics toolset Blender[21](version 2.91) for modelling and rendering. We used freely available textures and 3D mesh models for creating an initial database of objects. These objects include: *boxes, crates, racks, warehouse structures, forklifts, fire extinguishers* etc.

<sup>2</sup>*WareSynth* (Warehouse Dataset Generation Pipeline): <https://github.com/AnuragSahu/WareSynth>

### B. Generation process

Our generation process entails placement of objects in the scene procedurally in a randomized fashion, followed by adjustment of the lighting and textures. We perform texture editing and manipulate roughness and reflectance properties of objects to mimic real warehouse scenes.

We start with an empty warehouse. Racks are placed inside the warehouse according to a randomly generated 2D occupancy map. Lighting in the warehouse is also according to the same map, where we illuminate the corridors and also introduce ambient lighting. We keep the inter-shelf height and number of shelves in racks, width of corridors and overall rack density of the warehouse as parameters which can be tuned as per requirements. It is important to note that *WareSynth* is not constrained by our specific settings. The existing models can be readily substituted with custom box and rack models to configure the warehouse.

We also randomize the placement of boxes on each particular rack by specifying parameters which control the density of boxes placed and minimum distance between the boxes. We vary the density of rack occupancy by sampling the associated parameter from a uniform distribution between 0 (empty shelf) and 1 (fully occupied shelf). This ensures that the data is not imbalanced. Our algorithm picks a random box from available box models, and positions the same at a random angle varying between  $\pm r^\circ$  (where  $r$  can be specified). The boxes can also be stacked over each other, on individual shelves. This probabilistic procedure ensures that boxes are placed on the shelves randomly, but within the specified constraints, which helps us generate a large variety of realistic data.

### C. Data capture and annotation

We capture data via movement of a 6-DoF camera around the warehouse corridors by specifying a path or a set of discrete positions. The camera parameters can be varied in order to produce a diversity of views. The camera rotation, focal length, height above the ground etc. can all be manipulated and constrained according to the kind of views desired.

As per the requirement, we can capture the RGB images at the desired resolution for each of these camera positions, along with the camera intrinsic, and extrinsic parameters. We can also extract 2D annotations such as 2D bounding boxes and semantic and instance segmentation masks of the objects. By using this pipeline on a NVIDIA RTX 2080Ti we are able to generate 80 images per-minute. We can also obtain the 3D positions, orientations and 3D bounding boxes for all objects present in the camera FOV, along with depth maps and normal information. Our pipeline can also be used to obtain stereo-information. The obtained data can be easily be exported to various popular annotation formats such as KITTI, COCO, Pix3D, BOP etc.

### D. Applications and extensions

*WareSynth* can be used for various tasks such as 2D/3D object detection, semantic/instance segmentation, layout estimation, 3D scene navigation and mapping, 3D reconstruction

etc. The same pipeline can also be modified to other kinds of scenes such as supermarkets, greenhouses by changing the database of objects and placement parameters. The generation procedure and data capture methods are efficient, very flexible and can be customized as per user requirement. This makes the pipeline very useful for future research and generating annotated data at a large scale.

## IV. METHOD

### A. Problem Formulation

Formally, given a monocular color image  $\mathcal{I}$  of a warehouse rack in perspective view, we aim to predict the top-view (bird's eye view) layout for each shelf of the rack that lies within distance  $d$  from the camera (range of detection) and is visible in the image<sup>3</sup>.

Concretely, we wish to learn a labelling function that generates a top-view layout for the set of all scene points within a region of interest  $\Omega$ . Here, we consider  $\Omega$  to be a rectangular area around the concerned rack's center, in a top-down orthographic view of each shelf plane. The labeling function must produce labels for all the points in  $\Omega$ , regardless of whether or not they are imaged in  $\mathcal{I}$ . The points in  $\Omega$  are labelled as *occupied*, *unoccupied* or *background*. In our problem context, a pixel with *occupied* label denotes the presence of an object (boxes, cartons, crates etc.) at that place on the shelf, and the *unoccupied* label denotes that the shelf is empty at that pixel. Label *background* denotes the area that is not occupied by the shelf. As an additional task, we aim to learn a similar labelling function for the points in the front-view layout, which is orthogonal to the top-view layout. Here, we classify the empty inter-shelf area as *unoccupied*. Using a combined representation from these layouts, we obtain a 3D reconstruction of the rack, which can be further used for 3D spatial reasoning tasks. We discuss this extension in the further sections.

### B. RackLay Architecture

The architecture of RackLay comprises of the following subnetworks (refer Fig. 2):

- 1) A **context encoder** which extracts relevant 3D scene and semantic features from the input RGB image  $\mathcal{I}$  for layout estimation. This provides a context  $\mathcal{I}_e$  that helps us identify *occupied*, *unoccupied* and *background* scene points, for each shelf in the rack, in subsequent processing. We use a ResNet-18 encoder (pre-trained on ImageNet[22]) and fine-tune this feature extractor to learn low-level features that help reason across the three scene classes.
- 2) A **top-view decoder** that can comprehend the context to generate layouts for each shelf of the rack that is visible in the image. It decodes the context from the feature

<sup>3</sup>Flat-earth assumption: For the scope of this paper, we assume that the concerned rack is located within a bounded area in front of the camera, and that all planes(layers) in consideration (rack shelves, ground etc.) are more or less planar, with almost no inclination. The rack shelves are considered to lie on horizontal planes(layers) parallel to the ground plane.

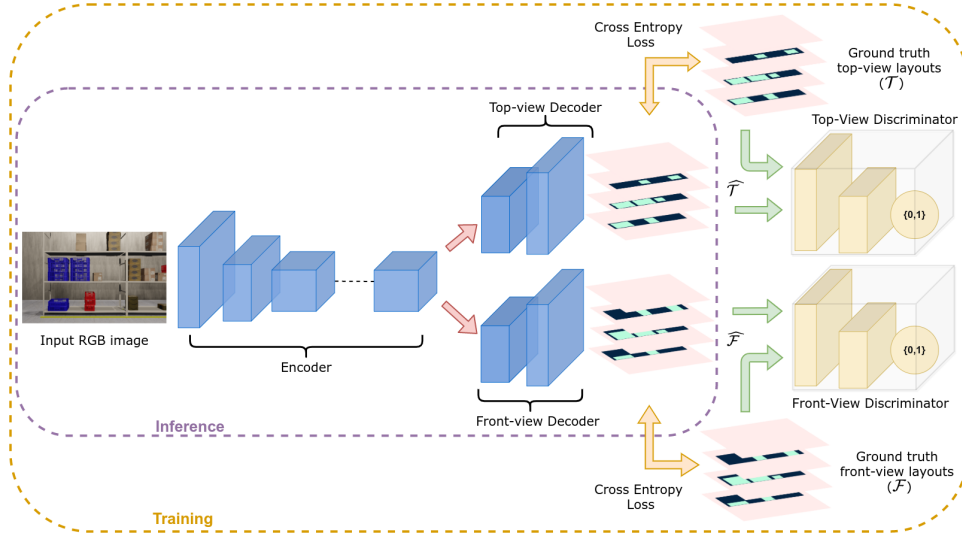


Fig. 2: The figure shows architecture diagram for *RackLay-D-disc*. It comprises of a context encoder, multi-channel decoders and adversarial discriminators. (Refer to Sec. V-B).

extractor (context encoder) via a series of deconvolution and upsampling layers that map the context to a semantically rich bird’s eye view. The decoder outputs an  $\mathcal{R} \times \mathcal{D} \times \mathcal{D}$  grid which represents the top-view layout  $\mathcal{T}$  where  $\mathcal{R}$  is the number of output channels and  $\mathcal{D}$  is the resolution for the output layouts. Each channel represents a shelf and is a per-pixel label map of the corresponding top-view layout. It is important to note the novelty of the associated design choice, i.e. using a multi-channel output to predict occupancy layouts which lie at different heights (layers) in the rack.

- 3) A **discriminator** is an adversarial regularizer. It refines the predicted layouts by regularizing their distributions to be similar to the true distribution of plausible layouts. The layouts estimated by the top-view decoder are input to this patch-based discriminator. The discriminator regularizes the distribution of the output layouts (*fake* data distribution, in GAN[23] parlance) to match a prior data distribution of conceivable scene layouts (*true* data distribution).

In order to deduce both views (top, front) from a single unified model, we extend the above architecture by adding an identical decoder to the existing encoder, followed by a discriminator, which predicts front-view layout for each shelf ( $\mathcal{F}$ ), just like the top-view layout ( $\mathcal{T}$ ).

### C. Formulation

We formulate the multi-rack shelf layout estimation as a multi-task probabilistic inference problem. Formally, let the top-view layout tensor over the domain  $\Omega$  be  $\mathcal{T}_{\Omega \times \mathcal{R}}$  and the front-view counterpart be  $\mathcal{F}_{\Omega \times \mathcal{R}}$ . We configure a deep neural network to maximize the posterior distribution  $P(\mathcal{T}, \mathcal{F} | \mathcal{I})$  given an RGB image  $\mathcal{I}$ .  $\mathcal{R}$  is a flexible parameter and denotes the maximum number of shelves the network can detect. Let  $\mathcal{T}_i$  and  $\mathcal{F}_i$  represent per-pixel occupancy label maps for the  $i^{th}$  shelf of the rack ( $i \in 1, 2, \dots, \mathcal{R}$ ).  $\mathcal{T}_i$

and  $\mathcal{F}_i$  are shelf-centric as well as aligned with the shelf’s coordinate frame. Conditioned on the encoded features  $\mathcal{I}_e$  of the input image  $\mathcal{I}$ , marginals  $\mathcal{T}$  and  $\mathcal{F}$  are independent of each other. Additionally, the components  $\mathcal{T}_i, i = 1, 2, \dots, \mathcal{R}$  and  $\mathcal{F}_i, i = 1, 2, \dots, \mathcal{R}$  are all independent of each other since the occupancy of each shelf does not depend on other shelves in the rack. Consequently, for the combined task, the posterior can be factorized as follows:

$$\begin{aligned}
 P(\mathcal{T}, \mathcal{F} | \mathcal{I}) &= P(\mathcal{T} | \mathcal{I}_e) P(\mathcal{F} | \mathcal{I}_e) \\
 &= P(\mathcal{T}_1, \mathcal{T}_2 \dots \mathcal{T}_{\mathcal{R}} | \mathcal{I}_e) P(\mathcal{F}_1, \mathcal{F}_2 \dots \mathcal{F}_{\mathcal{R}} | \mathcal{I}_e) \\
 &= \underbrace{\prod_{i=1}^{\mathcal{R}} P(\mathcal{T}_i | \mathcal{I}_e)}_{\text{top-view decoder}} \underbrace{\prod_{i=1}^{\mathcal{R}} P(\mathcal{F}_i | \mathcal{I}_e)}_{\text{front-view decoder}}
 \end{aligned}$$

### D. Loss function

The network parameters  $\phi, \psi, \theta$  of the context encoder, the top-view decoder and discriminator respectively are optimized using stochastic gradient descent.

$$\mathcal{L}_{sup}(\hat{\mathcal{T}}; \phi, \psi) = \sum_{j=1}^N \sum_{i=1}^{\mathcal{R}} f(\hat{\mathcal{T}}_i^j, \mathcal{T}_i^j)$$

$$\mathcal{L}_{adv}(\hat{\mathcal{T}}; \phi, \psi, \theta) = \mathbb{E}_{\theta \sim p_{fake}} [(\hat{\mathcal{T}}(\theta) - 1)^2]$$

$$\begin{aligned}
 \mathcal{L}_{discr}(\hat{\mathcal{T}}; \theta) &= \mathbb{E}_{\theta \sim p_{true}} [(\hat{\mathcal{T}}(\theta) - 1)^2] \\
 &\quad + \mathbb{E}_{\theta \sim p_{fake}} [(\hat{\mathcal{T}}(\theta) - 0)^2]
 \end{aligned}$$

where,  $\hat{\mathcal{T}}$  and  $\mathcal{T}$  are the predicted and the ground truth top-view layouts for each shelf,  $\mathcal{R}$  is the maximum number of shelves considered and  $N$  is the mini-batch size.

$\mathcal{L}_{sup}$  is the standard per-pixel cross entropy loss which penalizes deviation of the predicted layout labels ( $\hat{\mathcal{T}}$ ) from



their corresponding ground-truth values ( $\mathcal{T}$ ). The adversarial loss  $\mathcal{L}_{adv}$  encourages the distribution of layout estimates from the top-view decoder ( $p_{fake}$ ) to be close to the true data distribution ( $p_{true}$ ).  $\mathcal{L}_{discr}$  enforces the discriminator to accurately classify the network generated top-view layouts from the layouts sampled from the true data distribution. The discriminator loss  $\mathcal{L}_{discr}$  is the discriminator update objective[23]. Note that a similar set of loss terms exist for front-view layout estimation as well.

## V. EXPERIMENTS AND ANALYSIS

### A. RackLay Dataset

For the purpose of training and testing our network, using *WareSynth*, we generated 2 datasets, a simple dataset with 8k images and a complex dataset with 12k images<sup>4</sup>. We describe and display results for our more diverse and complex dataset consisting of 12k images, which we split into 8k/2k/2k for train/test/validation. We introduced two kinds of variations during data generation to add diversity and complexity in the scenes such that the resulting scenes mimic counterparts from real warehouses (see Fig. 5). We describe these variations below.

**Scene-level variation:** Each object placed on the racks is randomly chosen from a set of 6 distinct cardboard box categories and 2 different types of colored crates. These items all have different dimensions, textures and reflective properties (observe rows 1, 2 and 4 of Fig. 5). We also vary the inter-shelf height and the rack width between different scenes. The height up to which objects can be stacked over each other is also randomized (observe rows 2, 4 and 6 of Fig. 5). The background for the racks can be a wall (observe rows 1, 2 and 4 of Fig. 5) or other locations of a busy warehouse (observe rows 3, 5, 6, and 7 of Fig. 5), possibly containing other racks. Note that this setting poses a challenge for layout estimation since the network now needs to differentiate the concerned rack of interest from racks and other distractions in the background.

**Variation in camera placement:** The camera is placed such that it is directly facing the racks, and the image plane is orthogonal to the ground plane. For the camera, its horizontal distance to the rack and its vertical position above the ground plane are two parameters that are varied. Applying these variations affects the number of shelves visible in the image from 1 to  $\mathcal{R}$ . For our dataset, we set  $\mathcal{R}=4$  (observe rows 1, 2, 3 and 4 of Fig. 5).

**Ground-truth layout generation:** For every considered camera position, we record the corresponding ground truth information (location, dimensions, category) for all objects appearing within the intersection of the camera field of view (FOV) and considered range of detection as defined in the problem formulation (IV-A).

From this information, we generate ground truth top-view and front-view layouts by projecting onto the horizontal and vertical planes in the shelf-centric reference frame as described earlier. In our setting, the top-view and front-view

layouts are  $512 \times 512$  2D pixel grids (hence  $\mathcal{D} = 512$ ) which map to a corresponding  $8m \times 8m$  spatial extent. Therefore, the spatial resolution comes out to be 1.5625cm/pixel. This mapping helps us estimate free space in metric 3D.

### B. Evaluated Methods and Metrics

We evaluate the performance for the following approaches:

- *PseudoLidar-PointRCNN*: A PointRCNN based architecture [6], for 3D object detection on PseudoLidar [13] input, which involves converting image-based depth maps to LiDAR format. The 3D object detections are projected to the horizontal plane to obtain BEV layouts. We chose PointRCNN due to the success it enjoys in bird’s eye view 3D object detection tasks.
- *MaskRCNN-GTdepth*: Instance segmentation method [4], paired with ground-truth depth maps.

We compare that with following variants of RackLay:

- *RackLay-S*: Single decoder architecture, can be either for front-view *or* top-view.
- *RackLay-D* : Double decoder architecture, for both front-view *and* top-view.
- *RackLay-S-disc* : Single decoder architecture with discriminator, can be either for front-view *or* top-view.
- *RackLay-D-disc* : Double decoder architecture with discriminators, for both front-view *and* top-view.

We evaluate the layouts on both Mean Intersection-Over-Union (mIoU) and Mean Average-Precision (mAP). These metrics are calculated on a per-class basis.

### C. Results

We started with a standard encoder-decoder *RackLay-S* architecture, for predicting the top-view layouts. Having achieved superior results as compared to the baselines, we trained an identical architecture for predicting front-view layouts, which also gave similar results as top-view (refer Table I). To obtain both top and front views simultaneously in a single forward pass, we trained a double decoder model *RackLay-D* for estimating both top-view and front-view. Here, we observed gains in performance both quantitatively (refer row 1 in Table I) and qualitatively (refer to Fig. 3). We further improved upon *RackLay-D* with adversarial regularization to get our best network, *RackLay-D-disc*, which gives cleaner and sharper layouts, as discussed in ablation studies (refer Sec. V-E).

In Fig. 5, observe how our best network *RackLay-D-disc* is able to estimate layouts for a variety of scenes. For varying number of shelves (rows 1-4), we are able to predict layouts that are visible in the image and an empty layout for the rest. Our dataset also contains extremely sparse and densely packed shelves (rows 4-5), for which our network is able to reason out the thin spaces. We are also able to reason between the concerned rack and background clutter, the dataset contains images with and without background. Images with background clutter are shown in rows 3, 5, 6 and 7 and images with a wall behind are shown in rows 1, 2 and 4. Our network is also able to estimate layouts with for the case where no boxes are placed on a shelf as shown in row 7.

<sup>4</sup>Download RackLay datasets:<http://bit.ly/racklay-dataset>

Method	Top View				Front View			
	Rack		Box		Rack		Box	
	mIoU	mAP	mIoU	mAP	mIoU	mAP	mIoU	mAP
RackLay-D-disc	93.15	<b>98.73</b>	<b>95.07</b>	97.90	90.75	<b>98.54</b>	94.29	<b>97.95</b>
RackLay-D	<b>95.03</b>	98.37	92.94	97.63	<b>95.21</b>	98.48	<b>95.17</b>	97.94
RackLay-S-disc	92.34	98.28	93.71	97.85	91.96	98.13	92.65	97.51
RackLay-S	93.02	98.61	94.61	<b>98.07</b>	94.30	98.09	92.11	97.56
PseudoLidar-PointRCNN[6]	73.28	77.40	55.77	81.26	—	—	63.05	89.45
MaskRCNN-GTdepth[4]	36.48	42.48	35.57	47.44	—	—	—	—

TABLE I: **Quantitative results:** We benchmark the 4 different versions of our network- *RackLay-S*, *RackLay-S-disc*, *RackLay-D* and *RackLay-D-disc*, along with two baselines- *PseudoLidar-PointRCNN*[13], [6] and *MaskRCNN-GTdepth*[4] (as described in Sec. V-B). Note that *RackLay-S* and *RackLay-S-disc* are single decoder models and hence cannot predict top view and front view simultaneously. The top view and front view results displayed for each of these two models were trained separately (scaled results out of 100).

#### D. Comparison with baselines

1) *PseudoLidar-PointRCNN*: We perform 3D object detection using PointRCNN[6] on a PseudoLidar input[13]. The PointRCNN architecture was designed for detecting objects on a road like scene, their approach assumes a single dominant layer (the ground plane). This is an assumption used by many methods designed for bird’s eye view, and hence may not perform well for indoor scenes where multiple objects are scattered at different heights relative to ground plane. We observed that the success enjoyed by PointRCNN does not translate in the presence of multi-layer data. Their network is able to identify only the bottom shelf and objects kept on it. Therefore, we report metrics only for the bottom shelf layouts (refer Table I). This again highlights the importance of our work because we reason about bird’s eye view representation for multiple layers, rather than a single dominant layer.

2) *MaskRCNN*: We also compare with a classical approach wherein we use instance segmentation from MaskRCNN[4], and pair it with ground truth depth-maps. We project detected boxes to 3D shelf-wise, as boxes on a particular shelf will have similar vertical image coordinate. We then take a projection on the horizontal plane to obtain the box layouts for each shelf, by computing a convex hull for each box. Since this approach can only reason about visible points, it is clear from Table I that our network performs much better as it is able to perform amodal perception and is able to complete shapes and parts of the layout unseen in the input image. As discussed in Sec. I, MaskRCNN fails to predict segmentation maps for thin structures like shelves with good accuracy. Therefore, we present results only for box layouts.

#### E. Ablation studies

We conduct ablation studies to analyze the performance of various components in the pipeline. These lead to a number of interesting observations.

1) **Multi-task learning**: For the additional task of estimating the front-view layouts of the shelves, instead of using an identical single decoder model, we added a decoder to the existing encoder, which hence becomes the shared representation. Upon training our model for the dual task of front-view and top-view layout estimation, we observed that the losses were converging faster, with a slight quantitative

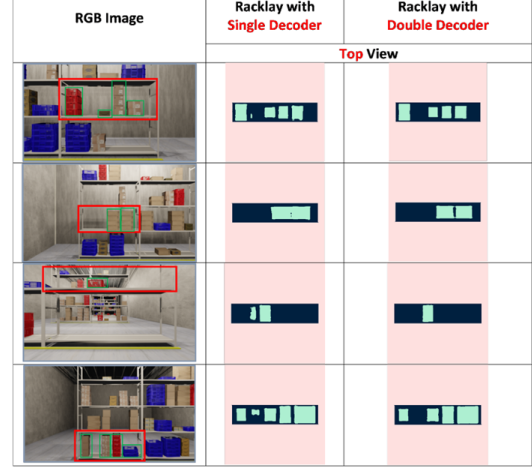


Fig. 3: **Effect of dual-task learning on top-view layout estimation:** The output layouts are being displayed only for the shelf bounded with a red box. The corresponding boxes for the bounded shelf are bounded with green boxes. Observe how the predicted boxes in **column 3** are more box-like and the presence of spurious noise in box predictions is lesser (**row 1 and 4**). The double-decoder model performs better in case where there are boxes very close to each other (**row 2**) and does a better job predicting the space between the boxes. It is also better at differentiating between boxes in the background and foreground (**row 3**). The single-decoder model confuses the background box to the left of the actual foreground box to be in the foreground as well, however the double-decoder model avoids this. The double-decoder model also avoids predicting spurious boxes as shown in (**row 4**).

improvement (refer Table I) in performance for almost all cases. There is a considerable improvement qualitatively as shown in Fig. 3, the network avoids predicting spurious boxes and outputs cleaner layouts. Making the network learn these two tasks together forces it to learn the relevant features related to the occupancy much faster and improves the performance metrics.

2) **Adversarial learning**: We add a discriminator at the end of the decoder to improve the layouts. At first glance, from Table I, using discriminators does not seem to produce any significant improvements quantitatively. However, there is considerable improvement qualitatively as shown in Fig 4, we obtain much sharper and realistic images. Most notably in the case of estimating layout for boxes, use of a discriminator reduces stray pixels which are mis-classified

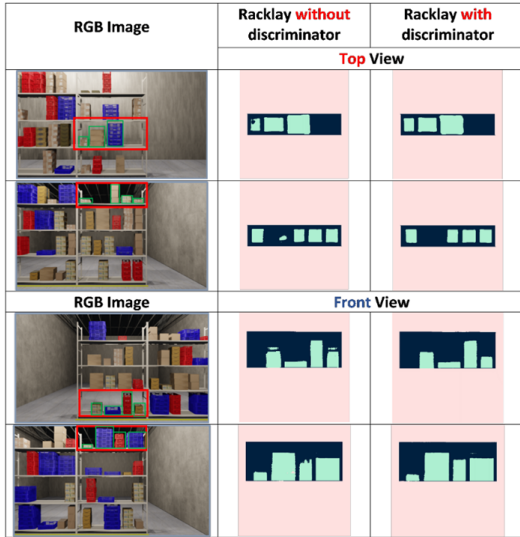


Fig. 4: **Effect of Discriminator on qualitative performance:** The output layouts in (columns 2-3) are being displayed only for the shelf bounded with a red box in (column 1). The corresponding boxes for the bounded shelf are bounded with green boxes. Observe how using a discriminator leads to more filled out layouts for boxes. The model without the discriminator tends to predict boxes with holes in them (rows 1, 3 and 4), whereas the discriminator ensures that this does not happen. The model with the discriminator also avoids false predictions for boxes as seen in row 2.

as boxes and outputs more clean box-like structures. Adding this component enhances the capability of the network to capture the distribution of *plausible* layouts.

#### F. Applications

1) **3D Free Space Estimation:** We first obtain the 2D bounding boxes of all shelves and each box kept on it, from the top-view and front-view layouts. Considering a particular shelf, we then combine the corresponding 2D bounding boxes from  $\hat{T}_i$   $\hat{F}_i$  to get respective 3D bounding boxes for each box stack on the shelf. Combining these representations for each shelf of a rack, we get a 3D volumetric reconstruction of the rack (refer Fig. 1). We then calculate the total capacity of a shelf from the inter-shelf height obtained in the front-view layout and subtract the volumes of the reconstructed object stacks to obtain the free volume available on the particular shelf (refer Fig. 5).

2) **Counting Number of Boxes:** After applying some morphological operations (either on top-view or front view), we get disjoint box layouts for each shelf. By counting the number of connected components, we compute the number of boxes kept on the particular shelf (refer Fig. 5).

## VI. CONCLUSION

We propose *RackLay*, which to the best of our knowledge is the first approach that provides multi-layered layout estimation, unlike previous approaches that assumed a single dominant plane. *RackLay*'s versatility is showcased across a large diversity of warehouse scenes and is vastly superior to prior art baselines adapted for the same task. We also release

a flexible dataset generation pipeline *WareSynth* that will aid future research for robotic tasks in warehouse scenarios.

## REFERENCES

- [1] P. Buxbaum, "Many warehouses don't have wms," Aug 2018. [Online]. Available: <https://www.globaltrademag.com/many-warehouses-dont-have-wms/>
- [2] "Trends in warehouse automation-2021," Oct 2020. [Online]. Available: <https://www.nitco-lift.com/blog/warehouse-automation-trends/>
- [3] O. Ronneberger and Fischer, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, 2015.
- [4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Maskrcnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.
- [5] T. Roddick, A. Kendall, and R. Cipolla, "Orthographic feature transform for monocular 3d object detection," *arXiv preprint*, 2018.
- [6] S. Shi, X. Wang, and H. Li, "Pointcnn: 3d object proposal generation and detection from point cloud," in *CVPR*, 2019.
- [7] C. Lee, V. Badrinarayanan, T. Malisiewicz, and A. Rabinovich, "Roomnet: End-to-end room layout estimation," *CoRR*, vol. abs/1703.06241, 2017.
- [8] H.-J. Lin, S.-W. Huang, S. Lai, and C.-K. Chiang, "Indoor scene layout estimation from a single image," *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 842–847, 2018.
- [9] A. Kar, S. Tulsiani, J. Carreira, and J. Malik, "Amodal completion and size constancy in natural scenes," in *International Conference on Computer Vision (ICCV)*, 2015.
- [10] S. Tulsiani, S. Gupta, et al., "Factoring shape, pose, and layout from the 2d image of a 3d scene," in *Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [11] J. Ku, M. Mozifian, et al., "Joint 3d proposal generation and object detection from view aggregation," in *IROS*, 2018.
- [12] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3d object detection," in *ECCV*, 2018.
- [13] Y. Wang, W.-L. Chao, et al., "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," in *CVPR*, 2019.
- [14] S. Gupta, J. Davidson, et al., "Cognitive mapping and planning for visual navigation," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7272–7281, 2017.
- [15] S. Schuster, M. Zhai, N. Jacobs, and M. Chandraker, "Learning to look around objects for top-view representations of outdoor scenes," in *ECCV*, 2018.
- [16] Z. Wang, B. Liu, S. Schuster, and M. Chandraker, "A parametric top-view representation of complex road scenes," in *CVPR*, 2019.
- [17] C. Lu and vande Molengraft, "Monocular semantic occupancy grid mapping with convolutional variational encoder-decoder networks," *IEEE Robotics and Automation Letters*, 2019.
- [18] K. Mani, S. Daga, et al., "Monolayout: Amodal layout estimation from a single image," *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.
- [19] C. Mayershofer, D. M. Holm, B. Molter, and J. Fottner, "Loco: Logistics objects in context," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2020, pp. 612–617.
- [20] "NVIDIA Isaac SDK," 2019. [Online]. Available: <https://developer.nvidia.com/isaac-sdk>
- [21] Blender Online Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Blender Institute, Amsterdam, Available: <http://www.blender.org>
- [22] J. Deng, W. Dong, et al., "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.
- [23] I. Goodfellow, J. Pouget-Abadie, et al., "Generative adversarial nets," in *Advances in Neural Information Processing Systems* 27, 2014.



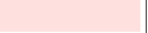
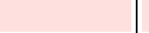
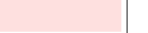

































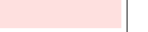





































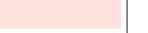












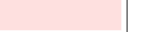





























RGB Input	View	Type	Output layout				Application	
			Shelf 1	Shelf 2	Shelf 3	Shelf 4	Prediction	Ground Truth
<b>1</b> 	<b>Top</b>	Predicted					4 Boxes	4 Boxes
		Ground Truth						
	<b>Front</b>	Predicted					23.77 m <sup>3</sup> Free Space	23.57 m <sup>3</sup> Free Space
		Ground Truth						
<b>2</b> 	<b>Top</b>	Predicted					7 Boxes	7 Boxes
		Ground Truth						
	<b>Front</b>	Predicted					41.88 m <sup>3</sup> Free Space	41.58 m <sup>3</sup> Free Space
		Ground Truth						
<b>3</b> 	<b>Top</b>	Predicted					12 Boxes	12 Boxes
		Ground Truth						
	<b>Front</b>	Predicted					56.90 m <sup>3</sup> Free Space	58.79 m <sup>3</sup> Free Space
		Ground Truth						
<b>4</b> 	<b>Top</b>	Predicted					16 Boxes	16 Boxes
		Ground Truth						
	<b>Front</b>	Predicted					77.36 m <sup>3</sup> Free Space	76.01 m <sup>3</sup> Free Space
		Ground Truth						
<b>5</b> 	<b>Top</b>	Predicted					7 Boxes	7 Boxes
		Ground Truth						
	<b>Front</b>	Predicted					83.92 m <sup>3</sup> Free Space	79.93 m <sup>3</sup> Free Space
		Ground Truth						
<b>6</b> 	<b>Top</b>	Predicted					9 Boxes	9 Boxes
		Ground Truth						
	<b>Front</b>	Predicted					73.25 m <sup>3</sup> Free Space	74.95 m <sup>3</sup> Free Space
		Ground Truth						
<b>7</b> 	<b>Top</b>	Predicted					2 Boxes	2 Boxes
		Ground Truth						
	<b>Front</b>	Predicted					90.22 m <sup>3</sup> Free Space	90.20 m <sup>3</sup> Free Space
		Ground Truth						

Fig. 5: **Diversity in dataset:** The output layouts are being displayed only for the rack bounded with a red box in **column 1**. Here, the boxes in both top-view and front-view are in green color, the free space is in dark blue color. The background is represented with pink. For the sake of visualisation, we have omitted the background class for visible shelves. Observe how our dataset contains images of varying number of shelves and boxes(**rows 1-4**), extremely sparse and densely packed shelves (**rows 4-5**) and scenes with background (**rows 3, 5, 6, and 7**) and without (**rows 1, 2 and 4**) background clutter.