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 the 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.
1 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 [4]. 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, 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 [18] or object detection [8]. 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) [17], warehouse rack shelves are disconnected and distinct planar segments present at multiple heights (layers) 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. 4), 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) Due to the scarcity and difficulty of annotating warehouse data, we open-source a flexible synthetic data generation pipeline WareSynth (Sec. 3) which enables the researcher/user to create and customize their own warehouse scenes and generate 2D/3D ground truth annotations needed for their task automatically. 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. 5.1.

(4) We show tangible performance gain compared to other baseline architectures [20] 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. 5).

(5) We also show that RackLay performs well on real-world data with the help of synthetic images generated using WareSynth (Sec. 5.6). We additionally release the RackLay real-world dataset consisting of 542 RGB images along with layout annotations of shelves and objects in both top and front view.

2 RELATED WORK

In recent years, learning scene layouts and obtaining volumetric representations directly from an RGB image has garnered a 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.

2.1 Indoor scene understanding

Room layout estimation from a single image [11, 13] is a popular problem in the context of indoor 3D scene understanding. There have also been a few approaches for amodal perception as well [9, 21]. Indoor scene understanding can rely on strong assumptions like a Manhattan world layout, which works well for single room layouts.
2.2 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 [10, 12] approaches combine information from images and LiDAR, others [17, 22] work by converting images to bird’s eye view representations, followed by object detection.

2.3 Bird’s eye view (BEV) representation
BEV semantic segmentation has been tackled mostly for outdoor scenes. Gupta et al.[7] demonstrate the suitability of a BEV representation for mapping and planning. Schulter et al.[19] 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.[23] build on top of [19] 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 [14], uses a variational autoencoder (VAE) to predict road layout from a given image, but only for the pixels present in the image. MonoLayout [15], 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.

2.4 Warehouse Datasets
There are very few datasets publicly available for warehouse settings. Real-world datasets like LOCO[16] 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 [1], 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. To this end, we introduce our dataset generation pipeline.

3 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 annotations.

3.1 Software and initial models
For modelling and rendering the warehouse setup, we used the open source 3D graphics toolset Blender[3](version 2.91). 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.

3.2 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 $\pm r^\circ$ about the vertical axis (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.

3.3 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.

3.4 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, etc., 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.

4 METHOD

4.1 Problem Formulation

Formally, given a monocular color image $I$ 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.

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 $I$. 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.

4.2 RackLay Architecture

The architecture of RackLay comprises of the following sub-networks (refer Fig. 2):

1. A context encoder which extracts relevant 3D scene and semantic features from the input RGB image $I$ for layout estimation. This provides a context $I_c$ 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[5]) 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 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 $\mathcal{R} \times D \times D$ grid which represents the top-view layout $\mathcal{T}$ where $\mathcal{R}$ is the number of output channels and $D \times D$ 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[6] 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 ($\mathcal{T}$).

4.3 Formulation

We formulate the multi-rack shelf layout estimation as a multi-task probabilistic inference problem. Formally, let the top-view layout tensor be $\mathcal{T}$ and the front-view counterpart be $\mathcal{F}$. We configure a deep neural network to maximize the posterior distribution $P(\mathcal{T}, \mathcal{F}|I)$ given an RGB image $I$. $\mathcal{R}$ is a flexible parameter and denotes the maximum number of shelves the network can detect. Let $T_i$ and $F_i$ represent $D \times D$ per-pixel occupancy label maps for the $i^{th}$ shelf of the rack ($i \in 1, 2, \ldots \mathcal{R}$). $T_i$ and $F_i$ are shelf-centric as well as aligned with the shelf’s coordinate frame. Conditioned on the encoded features $I_L$ of the input image $I$, marginals $\mathcal{T}$ and $\mathcal{F}$ are independent of each other. Additionally, the components $T_i$ and $F_i$ 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:

$$P(\mathcal{T}, \mathcal{F}|I) = P(\mathcal{T}|I_L)P(\mathcal{F}|I_L)$$

$$= P(T_1, T_2, \ldots T_{\mathcal{R}}|I_L)P(F_1, F_2, \ldots F_{\mathcal{R}}|I_L)$$

$$= \prod_{i=1}^{\mathcal{R}} P(T_i|I_L) \prod_{i=1}^{\mathcal{R}} P(F_i|I_L)$$

$$\text{top-view decoder} \quad \text{front-view decoder}$$

4.4 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.

$$L_{\text{adv}}(\mathcal{T}; \mathcal{F}; \phi, \psi, \theta) = \mathbb{E}_{\theta - p_{\text{fake}}} [ (\mathcal{T}(\theta) - 1)^2 ] + \mathbb{E}_{\theta - p_{\text{true}}} [ (\mathcal{T}(\theta) - 0)^2 ]$$

$$L_{\text{sup}}(\mathcal{T}; \mathcal{F}; \phi, \psi) = \sum_{j=1}^{\mathcal{R}} \sum_{i=1}^{\mathcal{R}} \mathcal{F}_j \{ \mathcal{T}_i \}$$

$$L_{\text{disc}}(\mathcal{T}; \mathcal{F}; \phi, \psi) = \mathbb{E}_{\theta - p_{\text{fake}}} [ (\mathcal{T}(\theta) - 1)^2 ] + \mathbb{E}_{\theta - p_{\text{true}}} [ (\mathcal{T}(\theta) - 0)^2 ]$$
where, $\tilde{T}$ and $T$ are the predicted and the ground truth top-view layouts for each shelf, $R$ is the maximum number of shelves considered and $N$ is the mini-batch size.

$L_{sup}$ is the standard per-pixel cross entropy loss which penalizes deviation of the predicted layout labels ($\tilde{T}$) from their corresponding ground-truth values ($T$). The adversarial loss $L_{adv}$ encourages the distribution of layout estimates from the top-view decoder ($\tilde{p}_{fake}$) to be close to the true data distribution ($p_{true}$). $L_{disc}$ enforces the discriminator to accurately classify the network generated top-view layouts from the layouts sampled from the true data distribution. The discriminator loss $L_{disc}$ is the discriminator update objective[6]. Note that a similar set of loss terms exist for front-view layout estimation as well.

5 EXPERIMENTS AND ANALYSIS

5.1 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. 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 $R$. For our dataset, we set $R=4$ (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, object type) 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 (4.1).

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

5.2 Evaluated Methods and Metrics
We evaluate the performance for the following approaches:

- PseudoLiDAR-PointRCNN: A PointRCNN based architecture [20], for 3D object detection on PseudoLiDAR [22] input, which involves converting image-based depth maps to LiDAR format. The 3D object detections are projected to the
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.

### 5.3 Results

We started with a standard encoder-decoder RackLay-S architecture, for predicting the front-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 1). 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 1) 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. 5.5).

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. 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.

### 5.4 Comparison with baselines

#### 5.4.1 PseudoLiDAR-PointRCNN

We perform 3D object detection using PointRCNN[20] on a PseudoLiDAR input.[22] 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 1). 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.

#### 5.4.2 MaskRCNN

We compare with a classical approach wherein we perform instance segmentation using MaskRCNN[8], and pair it with ground truth depth-maps. Leveraging the depth maps, we project the detected boxes to 3D, shelf-wise, using the fact that boxes on a particular shelf will have similar vertical image coordinate. We then re-project the obtained points on to the horizontal plane to obtain the top-view box layouts for each shelf, by computing a convex hull for each box. Since this approach can only reason about visible points in the image, it is clear from Table 1 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. 1, MaskRCNN fails to predict segmentation maps for thin structures like shelves with good accuracy. Therefore, we present results only for box layouts.

### 5.5 Ablation studies

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

#### 5.5.1 Multi-task learning

For the additional task of estimating the front-view layouts of the shelves, we added a decoder to
Figure 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).

5.5.2 Adversarial learning. We add a discriminator at the end of the decoder to improve the layouts. At first glance, from Table 1, using discriminators does not seem to produce any significant improvements quantitatively. However, there is considerable improvement qualitatively as shown in Fig 4, the network avoids predicting spurious boxes and outputs cleaner layouts. Most notably in the case of estimating layout for boxes, use of a discriminator reduces stray pixels which are mis-classified 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.

5.6 Results on Real-World Images

We test the transferability of our best network, Racklay-D-disc, from synthetic images to real-world images using a custom dataset of 542 images which are manually annotated in the required format. We divide this data into 442 images for training and 100 images for testing. The data is also augmented using methods such as color jitter and horizontal flip during training to increase the robustness of the model. We test the performance of the model on real world test images using 3 methods of training as described in Table 2:

(1) Using only synthetic data consisting of 8000 images (as explained in section 5.1).
(2) Using only real world data of 442 images.
(3) Using a combination of the above two training sets: train the model on synthetic data and then fine-tune it using the real-world train images.

Table 2: Results on real-world data: Here, we showcase the performance of Racklay-D-disc on real-world data using three different training methods. For the first method (row 1), we train the model purely on synthetic data. For the second method (row 2), we train the model purely on real-world data. In the third method (row 3), we train the model on synthetic data and finetune with real-world data. The scores are scaled out of 100.

Download real-world dataset: https://bit.ly/3BXvz4D
Figure 5: Diversity in dataset: The output layouts are being displayed only for the rack bounded with a red box in column 1. Here, the objects on the shelves 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.
The scores obtained using the first two methods are good but not at par with the results presented in Table 1. Using a combination of synthetic and real world data for training, the model performs the best and the scores are at par with the scores in Table 1. This not only shows that RackLay is able to transfer to real-world scenes effectively but also highlights the importance of the synthetic data generation pipeline. Due to the difficult nature of annotating the data, our synthetic data generation pipeline helps in generating large amounts of data which can be used to train the model. This model can then be finetuned with a comparatively smaller amount of real-world data to get good layouts. We present the results for a few real-world images trained using method 3 in the supplementary material (real_images.png).

5.7 Applications

5.7.1 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 $T^i$ and $F^i$ 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).

5.7.2 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).

6 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 vision-based tasks in warehouse scenarios.

REFERENCES

[1] 2019. NVIDIA Isaac SDK. https://developer.nvidia.com/isaac-sdk
[2] 2020. Trends in Warehouse Automation-2021. https://www.nitco-lift.com/blog/warehouse-automation-trends/
[3] Blender Online Community. [n.d.]. Blender - a 3D modelling and rendering package. Blender Foundation, Blender Institute, Amsterdam. http://www.blender.org
[4] Peter Buschbaum. 2018. Many Warehouses Don’t Have WMS. https://www.globallademag.com/many-warehouses-dont-have-wms/
[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In CVPR.
[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In Advances in Neural Information Processing Systems 27.
[7] Shubham Tulsiani, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. 2017. Cognitive Mapping and Planning for Visual Navigation. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017), 7272–7281.
[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. MaskRCNN. In Proceedings of the IEEE international conference on computer vision. 2961–2969.