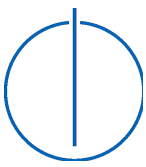
DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Data Engineering and Analytics

3D Instances from a single RGB Image

Peter Mortimer



DEPARTMENT OF INFORMATICS

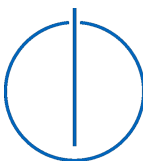
TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Data Engineering and Analytics

3D Instances from a single RGB Image

3D Instanzen von einem einzelnen RGB Bild

Author:	Peter Mortimer
Supervisor:	PD Dr. Ing. Habil. Federico Tombari
Advisor:	Yida Wang
Submission Date:	15.05.2019



I confirm that this master's thesis in data engineering and analytics is my own work and I have documented all sources and material used.

Munich, 15.05.2019 Peter Mortimer

Abstract

This master thesis compares different deep learning architectures for the task of 3D scene understanding from a single RGB image. The task of recovering 3D instances from a 2D image is challenging. The deep learning networks are required to estimate depth, shape, and pose of a 3D instance. This information is not directly provided in the RGB image input. We compare two different output representations for detected 3D instances.
One is a factored representation, where the instance shape is described in voxels and the instance's pose in separate scalars for the scale, rotation, and translation. The second representation encodes the shape and pose of each detected object directly in a point cloud.
The deep learning networks in this master thesis are trained on the SUNCG dataset, a synthetic dataset containing indoor scenes of houses. The comparison of the two output representations will be done on different quantitative and qualitative aspects.

Inhaltsangabe

In dieser Masterarbeit werden verschiedene Deep Learning Architekturen verglichen für den Anwendungsfall der 3D Objekterkennung von einem einzelnen RGB Bild. Das Problem der Erkennung von 3D Instanzen von 2D Bildern is anspruchsvoll. Die Deep Leaning Netzwerke müssen die Tiefe, Position, und Pose der 3D Instanz ermitteln. Diese Informationen sind nicht direkt im RGB Bild enthalten. Wir vergleichen zwei verschiedene Repräsentationen von 3D Instanzen.
Die erste Repräsentation teilt die Instanz in einzelne Faktoren. Die Form der Instanz wird mit einem Voxel Gitter beschrieben, während die Größe, Rotation, und Translation in getrennten Skalaren angegeben werden. Die zweite Repräsentation gibt die Pose und Form der Instanz direkt in einer Punktwolke an.
Die Deep Learning Netzwerke in dieser Masterarbeit werden auf dem SUNCG Datensatz trainiert, ein synthetischer Datensatz von Gebäuderäume. Der Vergleich der zwei Repräsentation erfolgt unter Betrachtung verschiedener quantitativer und qualitativer Aspekte.

Contents

1. Introduction

As humans we can be handed an image of a scene and even though we are not standing directly in the scene, but only looking at an image of it, we still have a good understanding of the contents of the scene and their relative position within the scene. This works especially well for environments we are familiar with, such as indoor scenes of homes. 3D scene understanding from a single RGB image tackles this type of problem and asks the more general question: What 3D information can be inferred from a single 2D image of scenes that one is familiar with?
The recent paper by Tulsiani et al. called "Factoring Shape, Pose, and Layout from the 2D Image of a 3D Scene" [Tul+18] trains an Artificial Neural Network (ANN) using a factored representation of each detected instance in indoor scenes. Their proposed ANN is trained on predicting the shape, scale, translation, and rotation for each of the detected objects separately. The predicted factors can then be combined to describe the object in relation to the camera.
Many of the current network architectures do not use a factored instance representation to describe the scene in 3D, but describe the complete scene in a 2.5D depth image [SNS09] or as voxels in a single voxel grid [Cho+16; Son+17]. Given only a single RGB image as input, we believe that the factored representation can provide a better predictive performance in differentiating between the objects in the scene and detecting their position and pose accurately within the scene. But we also believe that the factored representation leads to certain failure modes that can be corrected using more holistic instance representations, like 3D point clouds. In this work we further investigate the performance of the *Factored3D* network proposed by Tulsiani et al. [Tul+18] and see if this can be improved by the use of different output representations and loss functions. In particular, the factored predictions of the network can be combined and transformed to describe the object's shape and pose in relation to the camera in terms of a 3D point cloud. This allows us to leverage point cloud based loss functions to further improve the prediction performance for the factored output. We evaluate if this type of network fine-tuning can improve the different factors of the object prediction performance.

In this master thesis we make the following contributions.

1. We validate the reported performance of the proposed network in "Factoring

Shape, Pose, and Layout from the 2D Image of a 3D Scene" [Tul+18] by Tulsiani et al. using the SUNCG data set [Son+17].

2. We extend the network with a transformation of the factored output into a point cloud representation. This is all done using tensor operations in PyTorch [Pas+17] to preserve the gradient for backpropagation. This allows the network to improve the network components of the factored object prediction based on losses calculated on the point cloud representation of the object.
3. We compare the performance of the original factored prediction model by Tulsiani et al. and our proposed network extensions based on different quantitative and qualitative measures.

In the next chapter we will review current research related to our approach of using both a factored representation and a point cloud representation to predict object instances in a indoor scene using only a single RGB image.

2. Related Work

Our work is mostly based on the paper "Factoring Shape, Pose, and Layout from the 2D Image of a 3D Scene" from Tulsiani et al. [Tul+18] and in section 2.4 we put its important contributions in context to other factored representations used for scene understanding.
Apart from the aforementioned paper, our research is inspired by contributions from a few different areas of computer vision. Here we present these works and their approaches for the areas of 3D scene understanding, 3D object reconstruction from a single 2D image, deep learning on point cloud data, and factored representation of 3D instances.

2.1. 3D Scene Understanding

The objective in 3D scene understanding is to extract a rich but compact 3D representation of the physical space described in a high-dimensional input [Gei+]. Discoveries in 3D scene understanding are used for many indoor and outdoor applications, which include robot navigation [KAP13], mapping, localization, and autonomous driving [Che+16].
3D scene understanding can be categorized into static scene understanding for algorithms based on input images and dynamic scene understanding for video input data [NKP18]. In this work we focus on static scene understanding from a monocular view.

Most information we want to extract from image input is described by multiple neighboring pixels, which is why many scene understanding algorithms focus on extracting local geometric features. Convolutional Neural Network (CNN) have shown great success in different scene understanding tasks in recent years [SZ14; Gir15; Ren+15]. This moved the focus from handcrafted local features descriptors like HOG, SIFT, or SURF [DT05; Low99; Bay+08] towards learning-based approaches. The first CNNs were focused on 2D input and output data, which initially limited their use for 3D scene understanding tasks.

The emergence of larger 3D data sets allowed for the use of the previously mentioned learning-based approaches like ANNs also for 3D scene understanding. 3D data sets can be separated into synthetic and real data sets. The 3D data sets recorded from real world scenes, like the NYU-Depth V2 data set [NF12] or the Matterport3D V1 data set [Cha+17], offer realistic environments, but often lack in size (464 and 90 scanned scenes respectively) to be used to train an ANN.

In comparison, synthetic data sets like the SUNCG data set [Son+17] include over 45,000 scenes. The scenes were manually created on the Planner 5D platform [Pla] and resemble furniture layouts similar to real indoor scenes. The image quality of the SUNCG data set was greatly improved by the use of physically-based rendering [Zha+17], which creates a lighting environment more similar to real indoor scenes.

The larger synthetic indoor data sets like the SUNCG data set allowed for great progress for scene understanding algorithms based on a single depth image [Wan+18; Son+17; Fir+16]. Depth images as input resemble a realistic setup for robot navigation tasks [KAP13] and devices like the Microsoft Kinect show that good depth images can be created at a relatively low cost [Fir16]. The 2.5D depth images can be considered as 3D surface data. Each pixel in the depth image can be back projected to surface points in 3D camera space, when the intrinsic camera parameters are known.

SSCNet from Song et al. uses this 3D surface data from the depth image [Son+17]. SSCNet completes the full scene as a voxel grid, where each voxel entry is a categorical label for different furniture pieces found in the SUNCG data set. Synthetic datasets make scene completion feasible here by additionally providing accurate ground truth data for the occluded areas of the scene, that are unavailable to the depth camera. SSCNet uses 3D convolutional layers and a 3D extension of the dilated convolutional layers [YK16] to produce the semantic voxel grids.

The 3D voxel predictions tend to produce only very basic shapes for each detected object. The use of adversarial learning strategies, originally introduced by Goodfellow et al. [Goo+14], allow for a higher level of detail for the volume prediction both on the scene level [Wan+18] and on the object instance level [Yan+18b; Wu+16].

2.2. 3D Object Reconstruction from a single 2D Image

In an attempt to understanding a complete scene from a 2D image, the problem can be decomposed to the smaller tasks of recognizing the individual objects that make up the scene from the 2D image individually. This task of reconstructing a single object

compared to a full scene is often easier to solve. This led to a rapid development of deep networks for 3D object reconstruction and large 3D shape databases like ModelNet [Wu+] and ShapeNet [Cha+15] provided the necessary data to train deeper networks for 3D object reconstruction.

Many network architectures for 3D object reconstruction use a dense and regular 3D voxel grid to represent the 3D objects. In comparison to mesh and point cloud representations, the voxel grid representation has the most similar properties to 2D input images. Both voxel and image data are sampled from a equidistant grid and have a fixed size specified by their side length in each dimension. The voxel representation is therefore seen as the 3D analogue to the 2D pixel array [RUG17].

These networks use 3D convolutional layers and 3D pooling layers that have shown success for 2D data and fit them for the additional dimension [Gir+16; Son+17; Cho+16; Wu+15]. For instance, Gridhar et al. use 3D convolutional layers and 3D deconvolutional layers for their autoencoder-based network [Gir+16]. They use multiple training stages that alternate between 3D voxel input and 2D RGB input to produce a latent vector representation that is generative for 3D voxel data and predictable from 2D RGB input data.

The main shortcoming of dense 3D data is the cubic growth of the memory and computational requirements. This limits the output resolution of most voxel-based outputs to be around $32 \times 32 \times 32$, which does not allow for a great level of detail. Representing the dense 3D data in an octree data structure can allow for a higher resolution [RUG17; TDB17; Wan+17], but this is hard to incorporate into existing architectures.

2.3. Deep Learning on Point Cloud Data

There have been several advances in deep neural networks that process point cloud data. An object is thereby described by a set of $\{N_i | i = 1, ..., n\}$ points, where each point N_i is represented by a (x_i, y_i, z_i) triplet in 3D space, which for most application purposes represents an Euclidean space. A point cloud is most commonly represented as a $N \times 3$ matrix input for the deep neural network. A point cloud is orderless, meaning that a different order of the same points represents the same object. Deep neural networks for point clouds should therefore incorporate an invariance to permutation in their architectural design.

PointNet and its extension PointNet++ [Qi+17a; Qi+17b] apply pointwise feature

transformations with multi-layer perceptrons and global max pooling layers as a symmetric aggregation function to create a network that is invariant to permutation in the point set of the point cloud and robust to perturbations in the point cloud.

FoldingNet [Yan+18c] is an autoencoder trained on point cloud data. It uses graph-based layers to encode the point cloud in a latent feature vector and mimics the deformation of a 2D plane into a 3D surface in its decoder. The use of fully-connected layers to generate a point cloud from a latent vector representation has shown some success [Ach+18; Zam+18]. PCN [Yua+18] uses both decoding strategies to produce a coarse and detailed point cloud output for their task of completing a detailed point cloud from a sparse set of points. The very recent work by Zhao et al. introduces 3D-PointCapsNet [Zha+18], which uses latent capsules [SFH17; HSF18] as opposed to a single latent feature vector.

Overall, the use of deep neural networks on point clouds is promising. Point clouds allow for a simpler manipulation in terms of geometric transformations and deformations [FSG17], which is not as easily possible with dense 3D voxel grids. This can be leveraged by network architectures that take the properties of the 3D representation into account. Many works have shown that the chosen 3D representation for a specific computer vision task can greatly effect the model performance [SFH18; Tul+18].

2.4. Factored Representation of 3D Instances

There has also been also a lot of research on the use of alternative methods to describe 3D scenes. Primitives, such as cuboids [Tul+17; GEH10] and cylinders [BT73], were used in most early attempts to represent 3D scenes and objects. These simplified representations can often allow the training of more extensive scene properties like physical reasoning [Zhe+13].

Contrary to the more recent attempts to create end-to-end neural networks [Son+17; RPB15], there has been research done on models that predict independent factors, which later combined make up the final 3D representation [Tul+18; EW11]. A common reason to enforce a factored representation is to make the learned representation more understandable for us.

For instance, DC-IGN [Kul+15] is a deep autoencoder that learns a disentangled and semantically interpretable latent vector representation. The training procedure uses specific mini-batches with only one varying component, like the out-of-plane rotations or different lighting environments applied to the same object of interest,

while also clamping the change of other latent variables to enforce a specific training progress.

The *Factored3D* network by Tulsiani et al. [Tul+18] introduces a novel factored representation for the object prediction within a scene. The overall scene is factored into a background layout and a set of foreground objects. The foreground objects are also factored into separate components. Detected foreground objects by *Factored3D* are described by a normalized voxel grid and a separate prediction of the foreground object's scaling, translation, and rotation factors in relation to the camera position. The voxel grid is considered the 3D shape of the object, while the transformation represents the object's pose. The background layout is predicted in form of a 2.5D inverse depth map. The depth map prediction describes the amodal view of the scene. The amodal scene view describes the scene layout without any objects.

The most similar work to *Factored3D* is PoseCNN [Xia+18], which decouples the rotation and translation into separate network branches for 6D pose estimation. Both *Factored3D* and PoseCNN take a single RGB image as input and use object bounding boxes to make separate object predictions. Although PoseCNN assumes the 3D model of an detected object as already given, while *Factored3D* generates a new shape volume for each detected object.

In the following chapter we will introduce the *Factored3D* network proposed by Tulsiani et al. [Tul+18] in more detail. We also introduce the factored loss function used for training in the original paper in section 3.2 and our proposed extensions of using a point cloud based loss function in section 3.5. We also analyze the SUNCG data set used for training and evaluation to better understand the difficulty of the prediction task in section 3.4.

3. Methodology

The original *Factored3D* network proposed by Tulsiani et al. [Tul+18] makes many design decisions to successfully construct a 3D scene representation from a single 2D RGB image. In this chapter I will explain the individual design decisions in detail and mention the strengths and weaknesses that result for the network architecture based on them.
The difficulty of detecting the shape and pose of the SUNCG data set [Son+17; Zha+17] has not yet been adequately reviewed and presented. In section 3.4 we do this and evaluate the difficulty of the task of predicting the object shape and pose from the SUNCG data set.
In section 3.5 we introduce our extension to *Factored3D*, which converts the factored representation into a 3D point cloud.

3.1. Network Architecture

The goal of the *Factored3D* network is to predict the overall scene layout and a factored shape and pose prediction for each object present in the scene. The network architecture is illustrated in Figure 3.1. In subsection 3.1.1 we will explain the Layout module of the *Factored3D* network in more detail. The Coarse module, Fine module, and Bounding Box Module produce the factored shape and pose prediction and are explained in more detail in subsection 3.1.2.

3.1.1. Layout Prediction

Of the four network modules, the Layout module can be trained separately and is solely responsible for the depth prediction of the amodal scene layout. It takes an RGB image with a height of 128 pixels and width of 256 pixels as input and produces a one-dimensional inverse depth image with a height of 64 pixels and a width of 128 pixels. Each pixel value of the output image represents the predicted inverse depth from the camera in millimeters.

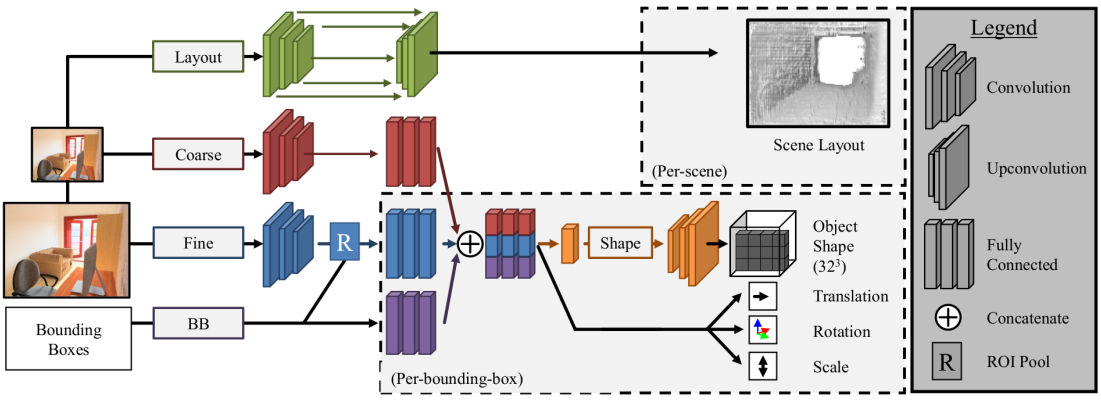


Figure 3.1.: Network Architecture of *Factored3D*. The network architecture of the *factored3d* network proposed by Tulsiani et al. [Tul+18] consists of 4 modules.
The Layout module creates an inverse depth image of the amodal scene based on a low-resolution version of the RGB input image.
The Coarse module recognizes global scene features based on a low-resolution version of the RGB input image.
The Fine module extracts detailed information from the high-resolution input image.
The area of the high resolution feature map of the Fine module, that corresponds to the proposed object bounding box, is passed on for object prediction as a fixed feature size through ROI-Pooling. The other proposed bounding boxes in the scene are passed on for object prediction encoded in the Bounding Box module.
The features from all three object prediction modules are concatenated and used in separate network branches to predict the shape code $\mathbf{s}$, the translation $\mathbf{t}$, the rotation $\mathbf{q}$, and scale $\mathbf{c}$.
The shape code $\mathbf{s}$ is a 20-dimensional latent feature vector, which is decoded into the $32 \times 32 \times 32$ shape voxel grid $\mathbf{V}$ by the shape decoder.

The architecture for the Layout module is based on the architecture of DispNet [NMa+16]. DispNet consists of a contractive part and an expanding part with long-range skip-connections [He+16] between them. DispNet was originally implemented for disparity estimation, but has shown sufficient results for its use in amodal scene depth estimation.

The contractive part of the Layout module consists of 6 convolutional blocks with 2 convolutional layers each. The second convolutional layer of each block has stride 2. The expanding part of the Layout module consists of 5 upconvolutional blocks, where each block consists of one upconvolutional layer followed by a convolutional layer. Features from the contractive part of the Layout Module and features from the upconvolutional layer preceding the convolutional operation of a block are passed on to the next block via skip layers.
This network architecture has been attributed to produce smooth output images due to the many skip layers [NMa+16]. This is useful for our task of creating depth maps of the amodal scene layout, which will mostly consist of straight walls, the floor of the room, and the ceiling of the room, which resemble very smooth depth changes in a depth map.
The Layout module is trained end-to-end using the L_1 objective. Given a inverse depth map prediction $\hat{\mathbf{H}}$ and a ground-truth inverse depth map $\mathbf{H}$, we define the following objective function.

$$\mathbf{L}_H = \|\mathbf{H} - \hat{\mathbf{H}}\|_1$$

3.1.2. Object Prediction

The main module for the factored object prediction is the Fine module, while the Coarse module and the Bounding Box module are considered contextual features by Tulsiani et al. [Tul+18]. The Layout module is trained on a scene level, where one training sample is one scene image, but the remaining modules for the object prediction are trained on a bounding box level. This means that one training sample is one bounding box in a scene and in most cases a scene image consists of multiple bounding boxes.

Fine Module. The Fine module takes a RGB image with a height of 480 pixels and a width of 640 pixels as input. The scene image is passed to the first 3 convolutional blocks of a ResNet-18 model [He+16] pretrained on the 1000-class ImageNet data set [Den+09; Ink]. The bounding box of the training sample is matched to the feature maps produced by the convolutional blocks and crops the feature map to the selected region of interest using a ROI-Pooling layer. The network therefore only passes features of the

high-resolution input image of the current bounding box of interest. These cropped image features are then passed to 2 fully connected layers of 300 units each.

Coarse Module. The Coarse module takes a RGB image with a height of 128 pixels and a width 256 pixels as input. The high-resolution input image is reshaped to fit into the low-resolution input dimensions of the Coarse module. The low-resolution scene image is passed to the first 4 convolutional blocks of a ResNet-18 model [He+16] pretrained on the 1000-class ImageNet data set [Den+09; Ink]. These are followed by 2 fully connected layers of 300 units each. The Coarse module provides features related to the global scene structure compared to the Fine module, which focuses on the object information found in the input bounding box. The feature map of the coarse module remains the same for each object proposal in the same scene.

Bounding Box Module. The Bounding Box module passes on all rectangle bounding boxes in the scene by passing two opposite corner points (x_1, y_1, x_2, y_2) for each bounding box. The bounding box coordinates are described in term of the high resolution input image.
The ROI-Pooling layer only receives the bounding box of the current object to predict. The Bounding Box Modules stores all bounding boxes in the scene, which allows for a better understanding of the object relationships in the input scene.
Depending on the training stage, which is further explained in section 3.3, the Bounding Box module uses ground-truth bounding boxes preprocessed by SUNCG data set or uses bounding box proposals generated from the RGB input image using Edge Boxes from Zitnick and Dollár [ZD14]. Edge Boxes is a bounding box proposal algorithm, which detects bounding boxes of objects in a scene based on the enclosed edges detected in an RGB image. It also returns a score $h_{bbox} \in [0, 1]$ for each bounding box, representing the likelihood of the bounding box to contain an object.
In Figure 3.2 the different bounding boxes are visualized for an example input image.

3.2. Factored Object Prediction.

The object prediction in *Factored3D* is done on the bounding box level. The features used for the factored object prediction are taken from the Coarse module, Fine module, and Bounding Box module (see Figure 3.1). Each module produces a 100-dimensional feature vector. These are concatenated to a 300-dimensional feature vector. At this point the *Factored3D* network branches to four separate paths for the shape, translation, rotation, and scale prediction. Each branch uses a linear layer to map the 300-dimensional

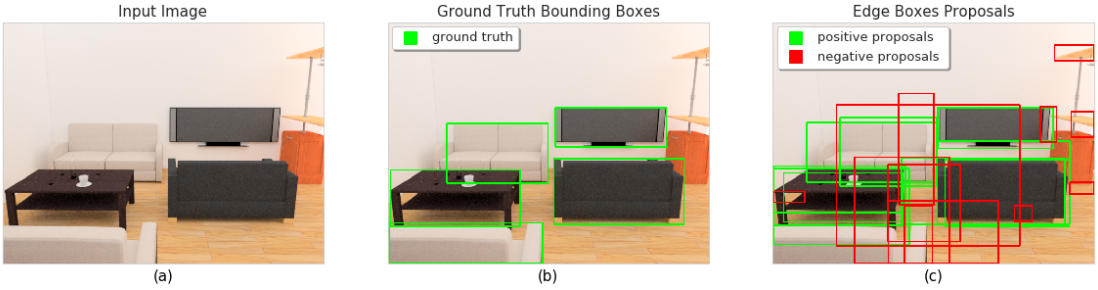


Figure 3.2.: Example Bounding Boxes. Image (a) shows a given training image from the SUNCG data set. The SUNCG data set has exact semantic information for the image content, which allows the accurate detection of the ground-truth bounding boxes. Image (b) displays the ground-truth of the input image in green.
We use Edge Boxes [ZD14] as an external source for object bounding box proposals. During training we distinguish between positive and negative bounding box proposals to train the model to tell whether a proposed bounding box does contain a foreground object or not. The construction of this loss function is explained in more detail in section 3.3.
A positive bounding box proposal has a bounding box IoU larger than 0.7 to any given ground-truth bounding box, while a negative bounding box proposal has a bounding box IoU smaller than 0.3. Bounding box proposals with IoU values between these ranges are considered inconclusive and are discarded before training. Image (c) displays a sample set of the positive and negative bounding box proposals suggested by Edge Boxes for our input image.

vector to the appropriate feature representation.

Shape Prediction. The final shape prediction of a detected object is described in terms of a $32 \times 32 \times 32$ voxel grid. The ground-truth voxel grid $\mathbf{V}$ of an object is discrete, where 1 means that the voxel is occupied by the object shape, while 0 describes an unoccupied voxel. The shape prediction $\hat{\mathbf{V}}$ is $32 \times 32 \times 32$ voxel grid with values in $[0, 1]$ range.
The pose of the object as it appears in the scene is not considered in the shape prediction, but the prediction is made in a canonical coordinate frame, where each object is represented front-facing and upright. This allows for less ambiguity during training of the network. This strictly separates the shape prediction from the pose prediction of an object.
A per-voxel cross-entropy loss is used as objective function during training.

$$\mathbf{L}_V = \frac{1}{N} \sum_n \mathbf{V}_n \log \hat{\mathbf{V}}_n + (1 - \mathbf{V}_n) \log(1 - \hat{\mathbf{V}}_n)$$

$\mathbf{L}_V$ is difficult to optimize without pretraining the voxel network layers to produce reasonable 3D shape outputs. Therefore, the shape decoder of *Factored3D* is initialized with the weights of a voxel autoencoder. The voxel autoencoder's architeciture is based on the T-L Network by Girdhar et al. [Gir+16]. The autoencoder is trained on the voxels of the objects available in the SUNCG dataset and has a 20-dimensional bottleneck vector to match the size of the shape code $\mathbf{s}$ in the *Factored3D* network. The architecture of the voxel autoencoder is illustrated in Figure 3.3.

Translation Prediction. The ground-truth translation vector $\mathbf{t}$ is a triplet $(\mathbf{t}_x, \mathbf{t}_y, \mathbf{t}_z)$ describing the object's position in the scene in camera space. The translation prediction $\hat{\mathbf{t}}$ is optimized using the squared Euclidean loss.

$$\mathbf{L}_t = \|\mathbf{t} - \hat{\mathbf{t}}\|_2^2$$

Rotation Prediction. The pose of the object is parameterized with a unit-normalized quaternion $\mathbf{q} = (\mathbf{q}_w, \mathbf{q}_x, \mathbf{q}_y, \mathbf{q}_z)$ transforming the object shape from the canonical coordinate frame in voxel space to its pose in camera space. Tulsiani et al. have shown in their design of the *Factored3D* model, that posing the rotation prediction as a regression problem does not perform well [Tul+18; Su+15; TM15]. They hypothesize that this problem might come from the mulit-modality when describing a rotation, especially of objects with symmetric shapes.
Therefore, the rotation prediction is reformulated into a classification problem. All rotations in the training set are clustered into 24 bins. The rotation prediction now produces a probability distribution $\mathbf{k}_d$ and is trained on the negative log likelihood loss.

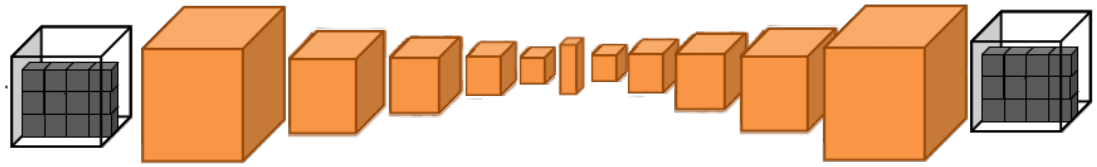


Figure 3.3.: Voxel Autoencoder. The voxel autoencoder is based on the T-L Network proposed by Girdhar et al. [Gir+16]. The bottleneck is a 20-dimensional latent vector to match the shape code $\mathbf{s}$ of the *Factored3D* network. The voxel autoencoder takes a $32 \times 32 \times 32$ voxel grid $\mathbf{V}$ as input.
The encoder consists of five 3D convolutional blocks, where each block consists of two 3D convolutional layers with a kernel size of 3, a stride of 1, and a padding of 1, followed by a 3D max pooling with a kernel size 2, a stride of 2, and a padding of 0. Two fully connected layers transform the 3D feature map to the 20-dimensional latent vector.
The decoder consists of five 3D upconvolutional blocks, where each block consists of a 3D upconvolutional layer with a kernel size of 4, a stride of 2, and a padding of 1, followed by a 3D convolutional layer with a kernel size of 3, a stride of 1, and a padding of 1. After the upconvolutional blocks comes one final 3D convolutional with a kernel size of 3, a stride of 1, and a padding of 1 before returning the autoencoder's reconstruction $\hat{\mathbf{V}}$ of the input voxel grid $\mathbf{V}$.
The autoencoder is trained for 800 epochs on the 2549 objects available in the SUNCG data set using a per-voxel cross-entropy loss.

The ground-truth rotation is converted to a one-hot encoded vector with an entry in the closest bin, which is defined as the superscript k in our expression.

$$\mathbf{L}_q = -\log(\mathbf{k}_d^k)$$

The quaternion with the highest bin entry after the rotation prediction is used for the final transformation of the object into camera space. The reformulation of the problem into a classification problem does show a better prediction performance on the SUNCG dataset, but this also restricts the overall predictive power of *Factored3D*. The *Factored3D* can only predict objects into these 24 poses, leading to large errors for objects in unusual poses that are not covered by these 24 bins. Furthermore, the network cannot easily be fine-tuned to learn new poses in transfer learning scenarios.
We believe for the problem domain of indoor scene understanding the rotation prediction as classification is not too restrictive. Indoor scenes contain many regularities, like roughly perpendicular walls and the alignment of objects along theses walls. These regularities in object poses are sufficiently captured by 24 different poses.

Scale Prediction. The ground-truth scaling vector $\mathbf{c}$ is a triplet $(\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_z)$ describing the anisotropic scaling applied to the voxel shape from voxel space to camera space. Similar to the translation prediction, the objective function poses the scale prediction as a regression task, but here we minimize the Euclidean loss between the scaling vectors in log-space.

$$\mathbf{L}_c = \| \log(\mathbf{c}) - \log(\hat{\mathbf{c}}) \|_2^2$$

3.3. Training Procedure

The object prediction of the *Factored3D* is trained in multiple stages (see Figure 3.4). Training the network in multiple stages improves the overall performance of the shape prediction, especially from bounding box proposals. The objective function changes slightly from one training stage to the next. The following subsections present the minor differences in the *Factored3D* network architecture and in the objective function in the three training stages.

3.3.1. Mimicking the Voxel Autoencoder from Ground-truth Bounding Boxes

The first training stage only trains the network based on ground-truth bounding boxes for each object in the scene (see Figure 3.4a). The shape prediction is also only limited

to transforming the 300-dimensional instance-specific bottleneck representation into the 20-dimensional shape code $\hat{\mathbf{s}}$, instead of decoding a full $32 \times 32 \times 32$ voxel shape. The predicted shape code $\hat{\mathbf{s}}$ is trained to be similar to the shape code $\mathbf{s}$ produced by the pretrained voxel autoencoder illustrated in Figure 3.3. The shape code prediction is trained using the L2 loss.

$$\mathbf{L}_V^{(l2)} = \sum_{i=1}^{20} (\hat{\mathbf{s}}_i - \mathbf{s}_i)^2$$

Each prediction factor also contains a hyperparameter α to weight the sum loss component accordingly. The object prediction is done on the bounding box level, where we denote b as a bounding box and $\mathcal{B}^{(gt)}$ as the set containing all ground-truth bounding boxes in the input image. This leads to the following objective function $\mathbf{L}^{(1)}$ for the first training stage of *Factored3D*:

$$\mathbf{L}^{(1)} = \sum_{b \in \mathcal{B}^{(gt)}} \alpha_V \mathbf{L}_V^{(l2)} + \alpha_t \mathbf{L}_t + \alpha_q \mathbf{L}_q + \alpha_c \mathbf{L}_c$$

3.3.2. Mimicking the Voxel Autoencoder from Bounding Box Proposals

The second training stage introduces the bounding box proposals produced by Edge Boxes [ZD14]. The network is now additionally trained to differentiate between bounding box proposals that contain a foreground object and proposals that only depict the background of the scene or a very small segment of a foreground object (see Figure 3.1b).
We denote the set of positive proposals as $\mathcal{B}^+$ and define every bounding box proposal b as a positive proposal if it has a 2D IoU larger than 0.7 with any ground-truth bounding box in the input image. $\mathcal{B}^+$ only contains proposals that enclose a large portion of a visible foreground object, which should allow for a good object prediction.
A bounding box proposal that has a 2D IoU smaller than 0.3 with every ground-truth bounding box in the input image is considered a negative proposal. We denote the set of negative proposals as $\mathcal{B}^-$ and these proposals only enclose a very small portion of a visible foreground object, which should lead to poor object predictions.
The bounding box proposals that aren't in $\mathcal{B}^+$ or $\mathcal{B}^-$ are considered ambiguous and are not used for training.

$$\mathcal{B}^+ = \{b | \exists b_{gt} \in \mathcal{B}^{(gt)} : \text{IoU}(b, b_{gt}) > 0.7\}$$
$$\mathcal{B}^- = \{b | \forall b_{gt} \in \mathcal{B}^{(gt)} : \text{IoU}(b, b_{gt}) < 0.3\}$$

The network is trained to predict from the 300-dimensional instance-specific bottleneck representation a foreground probability $f \in [0, 1]$. The foreground probability f represents the network's confidence whether the proposed bounding box is in $\mathcal{B}^+$ or $\mathcal{B}^-$. This is also called the *label loss* in the code by Tulsiani et al. [Tul+].
The foreground probability prediction is trained using a binary cross-entropy loss. This gives us the following objective function for the second training stage.

$$\mathbf{L}^{(2)} = \sum_{b \in \mathcal{B}^+} \alpha_V \mathbf{L}_V^{(l2)} + \alpha_t \mathbf{L}_t + \alpha_q \mathbf{L}_q + \alpha_c \mathbf{L}_c - \alpha_l \ln(f) + \sum_{b \in \mathcal{B}^-} -\alpha_l \ln(1 - f)$$

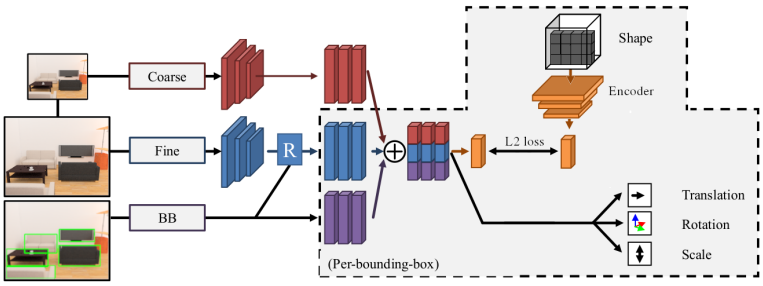
The foreground probability f is used as a threshold during inference to choose among the available bounding box proposals by Edge Boxes [Tul+18; ZD14].

3.3.3. Voxel Shape Prediction from Bounding Box Proposals

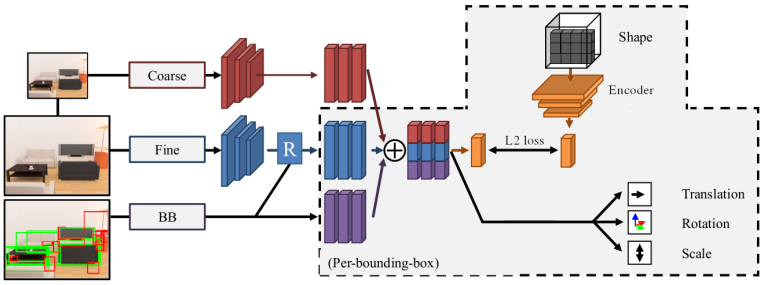
The third training stage incorporates the decoder component of the pretrained voxel autoencoder [Gir+16]. The decoder decodes the 20-dimensional shape code $\mathbf{s}$ into the $32 \times 32 \times 32$ voxel shape prediction $\hat{\mathbf{V}}$ (see Figure 3.4c). The decoder weights are initialized to the weights of the voxel autoencoder and then fine-tuned during training using a per-voxel cross-entropy loss.
The objective function only changes slightly from the previous training stage from a L2 loss $\mathbf{L}_V^{(l2)}$ based on the shape codes to a per-voxel cross-entropy loss $\mathbf{L}_V^{(3D)}$ based on the 3D voxel shapes.

$$\mathbf{L}^{(3)} = \sum_{b \in \mathcal{B}^+} \alpha_V \mathbf{L}_V^{(3D)} + \alpha_t \mathbf{L}_t + \alpha_q \mathbf{L}_q + \alpha_c \mathbf{L}_c - \alpha_l \ln(f) + \sum_{b \in \mathcal{B}^-} -\alpha_l \ln(1 - f)$$

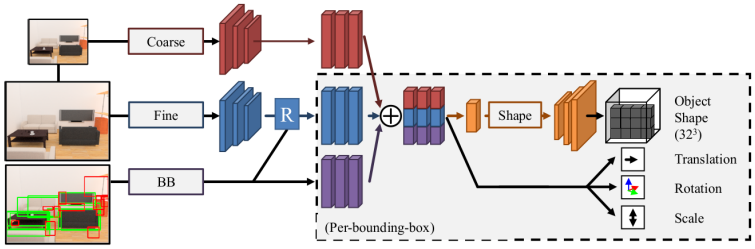
Overall, the multi-stage training procedure is necessary to create a full 3D scene prediction from just a single RGB image.
Stabilizing the network weights for reasonable shape predictions from imperfect bounding box proposals requires pretraining the network on ground-truth bounding boxes. Learning a full 3D shape prediction from scratch from 2D RGB images is not feasible in most cases, because the learned latent representation is not suitable for decoding into 3D space. Therefore, first mimicking the latent representation of the pretrained voxel autoencoder enforces the use of a latent representation suitable for 3D voxel decoding even though the latent representation is predicted from a 2D image.
On the other hand, the multi-stage training procedure makes the training procedure complex to reproduce and less flexible for slight changes in network components. For example, changing the pretrained voxel autoencoder used to support the shape prediction requires repeating all three training stages.



(a) First stage of training *Factored3D*. In the first training stage the network is solely trained on ground-truth bounding boxes. The shape prediction is trained to mimic the voxel autoencoder by producing the same shape bottleneck.



(b) Second stage of training *Factored3D*. In the second training stage the network is now also trained on bounding box proposals produced by Edge Boxes [ZD14]. Here we differentiate between positive bounding box proposals $\mathcal{B}^+$ that greatly overlap with ground truth bounding boxes and negative bounding box proposals $\mathcal{B}^-$ that do not contain any scene objects. *Factored3D* introduces a foreground probability f to indicate its confidence that the content of a given bounding box represents a scene object.



(c) Third stage of training *Factored3D*. In the third training stage the network adds the decoder component of the pretrained voxel autoencoder [Gir+16] to decode the voxel grid $\hat{\mathbf{V}}$ from the predicted shape code $\hat{\mathbf{s}}$. The decoder weights are fine-tuned during this training stage using a per-voxel cross entropy loss. The training is still done based on bounding box proposals provided by Edge Boxes [ZD14].

Figure 3.4.: Overview of the *Factored3D* object prediction training stages. *Factored3D* is trained over multiple stages to enable a reasonable 3D shape prediction from just 2D RGB information and to train the full factored prediction on bounding box proposals produced by Edge Boxes [ZD14].

3.4. SUNCG Data Set

The *Factored3D* network and our extensions are trained on the SUNCG data set [Son+17]. The SUNCG data set has been used to train and evaluate many 3D scene understanding networks due to its large size with over 40.000 indoor house models and accurate ground-truth data [Son+17; Tul+18; Wan+18; LVV18; Yan+18a].
In subsection 3.4.1 we give an overview of the SUNCG data set. We use the physicially-based renderings by Zhang et al. [Zha+17] for our training and evaluation. We therefore take a more detailed look at the viewpoint selection process used to create our training images.
Our training and test image set is analyzed on its value ranges for each factor in subsection 3.4.2. This analysis gives us a better understanding of the problem difficulty, as well as a comparison baseline for the original *Factored3D* network and our network extensions.

3.4.1. Overview

SUNCG is a large synthetic scene data set, which was originally introduced with SSCNet by Song et al. [Son+17]. The indoor layout, object alignment, and surface materials were designed by users of the Planner 5D platform [Pla]. This makes the rooms appearance fairly realistic while still having a large size with over 40,000 designed houses.
Thanks to the great work by Zhang et al., we have high resolution images with realistic lighting environments taken from different rooms in the SUNCG data set [Zha+17]. This was done using the physically-based renderer Mitsuba [Jak].

The choice of the position of the cameras for the images from Zhang et al. was based on the following criteria: a camera is placed at a random position in each room of a scene with a horizontal view direction within one of the six 60 degree wide horizontal bins. The camera has a random height in $[1.5\text{m}, 1.6\text{m}]$ range with a downward tilt of $11°$. A camera perspective is then selected if it contains at least 3 different objects, where each object covers at least 1% of the image's pixels. Additionally, the camera cannot be within 10cm of any scene obstacle to avoid unusual views that a physical camera could not recreate.
These conditions produce scenes that contain enough scene context to train a context-aware object prediction. Figure 3.5 shows a sample of the viewpoints selected for a given house of the SUNCG data set.

Factored3D was trained using the physically-based renderings by Zhang et al. [Zha+17], but limited the object prediction training to the six object categories: {bed, chair, desk,

Figure 3.5.: Viewpoint Selection for Physically-Based Renderings. This visualization by Zhang et al. [Zha+17] shows the selected viewpoint positions for a given house model of the SUNCG data set [Son+17] with the floor plan of the house. Each selected viewpoint contains at least 3 objects that take up 1% of the image's pixels. The chosen viewpoint positions are realistic viewpoint arrangements for images taken by humans of indoor scenes.

sofa, table, television} [Tul+18]. The viewpoint selection process from Zhang et al. did not put a restriction on these six object categories, which means that a few input images might not include any objects for *Factored3D* to predict. Our proposed network additions are also trained under these conditions.

3.4.2. Factor Variations in the SUNCG Data Set

Aside from the *Factored3D* network, there is no work that uses the SUNCG data set to train a factored object prediction. There is not a lot of past research to rely on that show the suitability of using the SUNCG data set for evaluating scene understanding tasks. We quantify the difficulty of a factored prediction on the physically-based renderings of the SUNCG data set by creating comparison baselines. These comparison baselines provide the necessary context to correctly evaluate the results of the *Factored3D* network and our network extensions in chapter 4.

We limited our comparison baselines to the shape and scale factors. The rotation factor is classified into 24 medoid bins, which already distributes the objects among the rotational bins very well. The translation factor also does not show any regularity in terms of the object depth from the camera that warrants a comparison baseline.

Shape Baseline. Due to the inherent uncertainty of predicting the precise geometry of an object, ANNs tend to produce a mean shape to average out the space of uncertainty [FSG17]. This mean shape only contains the most basic aspects of the object shape. We view this as a lower bound of the expected object prediction performance and create a baseline by producing surrogate mean shapes for each of the six object categories.
Our surrogates for the mean shapes are produced by taking the most common occupied voxels across each of the six object categories. We sum up the voxel shapes of all objects in the same category. Each voxel in the voxel grid has a value representing the number of object shapes that occupy the same voxel position. We can then produce iso volumes by choosing voxels that appear in at least n objects within the category. We choose the value n^* that produces a volume with a total number of voxels closest to the median number of voxels for the object category.
The six mean shapes are illustrated in Figure 3.6 together with three example instances from each object category.

Scale Baseline. The variance between the scale values for each object category is rather small, which could lead to scale predictors performing well by just predicting median scales (see Figure A.1 in the appendix for more details). We therefore create a baseline scale predictor, that for a given object instance predicts the median scale for the given object category in the SUNCG data set.

Figure 3.6.: Mean Shapes as Comparison Baseline. The first three columns show example instances from each object class. The fourth column presents the mean shape produced by taking the most common voxels within each category. The number of selected voxels corresponds to the median number of voxels that an object in the category occupies.
The mean shapes only resemble a very primitive shape description of the given object category and often lack important details (mean shape table and chair both do not have any legs). We use the mean shapes as a handcrafted comparison baseline to evaluate the shape prediction of *Factored3D* and our proposed network extensions (see Table 4.1).

3.5. Point Cloud Transformation

Our largest addition to the *Factored3D* network is a point cloud transformation from the factored object prediction and this is presented in this section. First we explain our motivation for adding a point cloud transformation in subsection 3.5.1. This is followed by a detailed description of our implementation of the point cloud transformation in subsection 3.5.2, where we also reason our design decision made for our implementation. Finally, we go into detail on the point cloud based loss functions we will use to train our extended *Factored3D* network on in subsection 3.5.3.

3.5.1. Motivation

We believe that the factored component prediction of a scene object and a more holistic object prediction are both promising approaches for solving scene understanding tasks, but each approach has different strengths and weaknesses. We will first mention certain failure modes that can appear with the factored representation used by *Factored3D* and then introduce a 3D point cloud representation as a holistic object instance representation.

Factored3D can learn fairly good scene predictions by separating the pose and shape prediction into simpler subtasks. Although the shape and pose information is represented in a unified way to *Factored3D* during training with canonical coordinate frames for the shape description and a fixed voxel grid density, there still exists natural ambiguities when predicting scene objects in terms of factored components.

For instance, a table with a symmetric shape could be predicted with a rotation off by 180°. The composed object prediction would be satisfactory, since the symmetric shape does not indicate a unique pose, but *Factored3D* would train to correct the large rotation error of 180° from the ground-truth quaternion. Tulsiani et al. report spikes in the rotation error for angle distances around 90° and 180° from the ground-truth rotational pose for objects and they attribute this to these natural ambiguities [Tul+18].

Another source for ambiguities in the factored representation can come from the relationship between the shape description and the scale prediction. An object like a bed with few granular shape details could be described with very few voxels in the shape voxel grid and describe its cuboid shape with the matching anisotropic scale factor. But the same bed could be described using more voxels in the shape voxel grid and a slightly smaller scaling factor. The final object prediction of both beds would be satisfactory, but the factored description is again ambiguous.

These failure modes of the factored representation motivated us to investigate if extending the *Factored3D* with a holistic object prediction could improve the overall object prediction performance of the factors.

Representing an object in terms of a 3D point could follows a more holistic approach. The shape, translation, rotation, and scale of an object can be directly encoded by the 3D point cloud of an object. Point clouds can be more easily manipulated using geometric transformations [FSG17]. This is not the case when a shape is described in a voxel grid with a separate pose description.
These observations motivated us to test if using a point cloud based representation of the final factored output coupled with a point cloud based objective function can lead to improvements in the *Factored3D* architecture for the object prediction.

3.5.2. Implementation

The factored object prediction encodes all the necessary information to describe the voxel shape as a point cloud in world camera space. We want to describe the transformation in terms of tensor operations to preserve the automatic differentiation provided by PyTorch [Pas+17] to allow the loss calculated on the point cloud representation to pass backward and update the weights within the factored prediction of *Factored3D*.

The shape predictor produces float values in $[0, 1]$ range, which we threshold using $\delta_{vox} = 0.25$ to occupied and empty voxel entries. The occupied voxel values are masked to a vector v_{ones} with only 1-entries, which is multiplied with a vector containing the corresponding 3D point cloud index (x, y, z) for each occupied voxel entry. The index vector defines the 3D point to be in the center of the occupied voxel and the whole object in voxel space is centered around the origin with the voxel grid spanning $[-0.5, +0.5]^3$. We define the resulting point cloud in voxel space as P_{vox}.

The point cloud P_{vox} does not have a fixed number of 3D points across all object predictions, since the number of occupied voxels for each object shape varies. We randomly sample all point clouds P_{vox} to a fixed point size $k = 1024$. This allows us to store all point clouds of a batch into a tensor without any padding. Some point cloud based distance measures, like the Earth Mover's distance, require the point clouds to be of equal length. The mean number of occupied voxels for the objects of our six selected categories in the SUNCG data set is 4070. The number of predicted voxels will typically be lower for our shape predictions, but $k = 1024$ are enough points to still preserve the structure of the full point cloud.

The translation, rotation, and scale prediction are composed to a homogeneous trans-

formation matrix T_{cam}. T_{cam} transforms 3D points in voxel space to 3D points in camera space. We apply T_{cam} to our point cloud in voxel space P_{vox} to transform our point cloud object to its predicted position in camera space, denoted as P_{cam}.

3.5.3. Loss Functions for Point Clouds

The lack of order and the non-uniform point density of 3D point clouds require the use of different loss functions compared to 3D voxel grids, which can often use 3D extensions of 2D loss functions like binary cross-entropy for training. The field of using deep learning on point cloud data is still relatively new. So far two loss functions are commonly used: the Chamfer distance and the Earth Mover's distance [Yan+18c; Gro+18; FSG17; Yua+18; Zha+18; Ach+18]. In the following we will briefly introduce both the Chamfer distance and the Earth Mover's distance.

Chamfer Distance. The Chamfer distance has originally found use for matching detected points in 2D image space [Bar+77]. This can be extended to 3D space. The Chamfer distance $D_{CD}(S_1, S_2)$ is calculated by summing up the shortest Euclidean distance from every element in S_1 to any element in S_2 and vice versa. The original defintion of the Chamfer distance adds both sums [Bar+77], but we used the extended Chamfer distance which returns the average (non-squared) Euclidean distance of the larger sum as used for FoldingNet [Yan+18c].

$$D_{CD}(S_1, S_2) = \max \left\{ \frac{1}{|S_1|} \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2, \frac{1}{|S_2|} \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2 \right\}$$

The Chamfer distance is differentiable and can be computed efficiently in $O(n \log n)$ time [Yua+18]. It is fairly robust to outliers, since we sum all closest matchings for each point cloud. The Chamfer distance does not require equally sized point clouds $|S_1| \neq |S_2|$ and multiple multiple points in one set can be matched to the same point in the corresponding set.
We use an implementation of the Chamfer distance in PyTorch provided by Christian Diller [Dil].

Earth Mover's Distance. The use of the Earth Mover's distance for measuring shape similarity was first proposed by Mumford [Mum91]. Informally speaking, the Earth Mover's distance can be understood as the amount of work needed to transform one probability distribution into another with the same integral [Cab+08]. We use an approximation of the original Earth Mover's distance that can be computed with a

runtime complexity of $O(n^2)$ [Ber85; Yua+18].

$$D_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \frac{1}{|S_1|} \sum_{x \in S_1} \|x - \phi(x)\|_2$$

The function $\phi : S_1 \to S_2$ is a bijection that minimizes the distance between matched points. The matching function ϕ is a bijection, which therefore requires equally sized point clouds $|S_1| = |S_2|$. The higher runtime complexity of calculating a differentiable approximation of the Earth Mover's distance requires us to randomly subsample our point clouds to smaller sets of size $k = 512$.
The Earth Mover's distance property of creating a one-to-one matching between each point in each 3D point cloud could lead to a more accurate expression of error in terms of the pose and shape. In comparison, the simpler Chamfer distance measure can be applied to two sets of different sizes, because many points in one set can be closest to the same point in the other set.
We use an implementation of the approximate Earth Mover's distance in PyTorch provided by Marc Eder [Ede].

4. Experiments

In this chapter we describe the experiments done to evaluate the prediction performance of the original *Factored3D* network and compare it with our proposed network extensions. We focus both on the predictive performance of the individual objects in a scene in section 4.2 and on the predictive performance of the whole scene considering all foreground objects at once in section 4.3. For each evaluation we will introduce our proposed evaluation metrics and discuss their meaning based on our results. We will also qualitatively evaluate the predictive performance of our networks on a real indoor scene data set in section 4.4. We introduce the different data sets used for evaluation in section 4.1.

4.1. Data Sets for Evaluation

All networks reported for the object level evaluation in section 4.2 and the scene level evaluation in section 4.3 are trained on the SUNCG traning set. We split the 41497 houses in the SUNCG data set, using 75% (31123) as a training set for the networks, 15% (6225) as a validation set for hyperparameter tuning, and 10% (4149) as a test set for evaluation. Our split of the SUNCG data set differs from the split used by Tulsiani et al. [Tul+18]. This is to see how well *Factored3D* performs on a different training set. The SUNCG data set contains a total of 2549 different objects with varying surface materials, but we limit our object prediction to the six object classes: {bed, chair, desk, sofa, table, television}. The original object predictor of *Factored3D* was also trained under these experiment conditions. The six object classes contain 845 different objects in total.

For the qualitative evaluation on real indoor scenes in section 4.4 we use the RGB images of the NYU-Depth V2 data set [NF12]. Here we only take a few example input images and use Edge Boxes [ZD14] to determine bounding box proposals for our networks.

4.2. Object Level Evaluation

In this experiment we compare the performance of different object prediction networks on the ground-truth bounding boxes of the SUNCG test set (see Table 4.1). All proposed networks are built on top of the three training stages of *Factored3D* presented in section 3.3. We also compare the performance to our handcrafted baselines for the factored object prediction on the SUNCG data set, which where introduced in subsection 3.4.2.

4.2.1. Evaluation Metrics

These evaluation metrics were proposed by Tulsiani et al. [Tul+18] to consider a prediction successful for the given factor.

Shape. A shape prediction is considered successful if the IoU of the predicted 3D voxel shape $\hat{\mathbf{V}}$ and the ground-truth voxel shape $\mathbf{V}$ is larger than $\delta_V = 0.25$.

Rotation. An object's rotation is considered successfully predicted if the geodesic distance between the predicted rotation $\hat{q}$ and the ground-truth rotation q is smaller than $\delta_q = \frac{\pi}{6}$, which corresponds to an angular distance of 30° or less. The geodesic distance Δ_q between two quaternions can be calculated in the following way by transforming the quaternions $\hat{q}$ and q into rotation matrices $\hat{\mathbf{R}}$ and $\mathbf{R}$ and using the Frobenius norm $\|\cdot\|_F$:

$$\Delta_q(\mathbf{R}, \hat{\mathbf{R}}) = \frac{1}{\sqrt{2}} \| \log(\mathbf{R}^T \hat{\mathbf{R}}) \|_F$$

All models we evaluate pose the rotation as a classification problem, where an object's orientation is classified into one of the 24 bins. We compare the corresponding quaternion of the predicted bin with the ground-truth rotation for the object.

Translation. An object's predicted position in camera space is considered successfully predicted if the predicted translation $\hat{\mathbf{t}}$ and the ground-truth translation $\mathbf{t}$ are within 1m from each other ($\delta_t = 1$). The camera space is also a Euclidean space, so the Euclidean distance can be used for calculating the translational error: $\Delta_t(\mathbf{t}, \hat{\mathbf{t}}) = \|\mathbf{t} - \hat{\mathbf{t}}\|_2$.

Scale. The distance between the predicted scale $\hat{\mathbf{c}}$ and the ground-truth scale $\mathbf{c}$ is measured using the logarithmic difference for each factor:

$$\Delta_c(\mathbf{c}, \hat{\mathbf{c}}) = \frac{1}{3} \sum_{i=1}^{3} |\log_2 \mathbf{c}_i - \log_2 \hat{\mathbf{c}}_i|$$

The threshold for a successful scale prediction is $\delta_c = 0.5$, which corresponds to the predicted scale being within a factor of $\sqrt{2}$ to the ground-truth scale.

4.2.2. Network Overview

In Table 4.1 we compare five different training procedures excluding the handcrafted baseline. The handcrafted baseline was generated as a comparison to the overall performance of the *Factored3D* model and our related network extensions (see subsection 3.4.2 for more details).
Please note that each training procedure is applied on top of the *Factored3D* training procedure presented in section 3.3. In the following we present the five training procedures we compared in Table 4.1:

- Factored3D: This is our network baseline and represents training *Factored3D* for an additional epoch in the third training stage, where the voxel decoder is fine-tuned on the bounding box proposals (see subsection 3.3.3).
- Factored3D + D_{CD}: Here the network is trained on the factored losses as usual for *Factored3D*, but we also add the Chamfer distance loss D_{CD} on the combined point cloud representation of the object. This is also trained for one epoch.
- Solo D_{CD}: The network is trained for one epoch only using the Chamfer distance loss on the full point cloud representation in camera space.
- Factored3D + pose-only D_{CD}: The network is trained using the standard factored loss components for the scale and translation prediction, but the shape and rotation is trained using the Chamfer distance loss on the rotated point cloud shape in object space.
- Factored3D + pose-only D_{EMD}: The network is trained using the standard factored loss components for the scale and translation prediction, but the shape and rotation is trained using the Earth Mover's distance on the rotated point cloud shape in object space.

4.2.3. Results

As seen in Table 4.1, the overall performance of the factored prediction on each factor does not vary too much. This is to be expected, since the different training strategies resemble a fine-tuning of already well-trained weights of *Factored3D*.

	Shape		Rotation		Translation		Scale	
method	%(IoU > 0.25)	Med-IoU	%(Err < 30)	Med-Err	%(Err < 1m)	Med-Err	%(Err < 0.5)	Med-Err
Factored3D	47.80	0.19	**78.10**	5.20	**92.50**	0.27	88.30	0.11
Factored3D + D_{CD}	**48.00**	0.19	77.50	5.27	92.20	0.28	86.00	0.15
Solo D_{CD}	46.30	0.17	76.60	5.60	92.30	0.28	51.00	0.49
Factored3D + pose-only D_{CD}	47.00	0.16	68.40	9.70	91.20	0.30	88.50	0.11
Factored3D + pose-only D_{EMD}	47.10	0.18	75.30	5.87	91.40	0.31	**88.80**	0.11
handcrafted baseline	**63.89**	0.35					70.87	0.27

Table 4.1.: Performance predictions on the ground-truth bounding boxes of the SUNCG test set. In this table we report the predictive performance for the individual factors. The predictions are made using the ground-truth bounding boxes for all objects in the SUNCG test set. We report the median for each factor metric and the percentage of the object predictions that have an error below/a performance above our defined thresholds. The *Factored3D* performance reported by Tulsiani et al. may vary due to the use of a different split of the SUNCG data set for training and testing [Tul+18].

The largest performance decrease can be observed for the scale prediction. A network fine-tuned only using the Chamfer distance loss performs very poorly in the scale prediction (51.00% of the test set reaches the scale threshold) compared to the standard *Factored3D* training strategy (88.30% of the test set reaches the scale threshold). When analyzing the training progress of a network solely trained on the Chamfer distance loss, we often observed the behavior, where the predicted point cloud is shrunk to a smaller scale and moved into the ground-truth point cloud to be fully enclosed by it. Numerically this decreases the Chamfer distance, but does not improve our overall object prediction performance. We believe that this kind of training behavior is also present here in the poor scale prediction performance.

To resolve this issue when using a point cloud based loss in camera space, we experimented only applying the point cloud-based losses on point clouds rotated in object space instead of introducing their translation and scale as the object appears in camera space. Only the predicted shape and rotation from the *Factored3D* prediction is passed to our point cloud transformation procedure, while the scale and translation are still trained using the factored loss functions. We observe a similar performance to *Factored3D* for all factors (Shape: 47.80%, Rotation: 78.10%, Translation: 92.50%, Scale: 88.30%), both when using the Chamfer distance (Shape: 47.00%, Rotation: 68.40%, Translation: 91.20%, Scale: 88.50%) and the Earth Mover's distance (Shape: 47.10%, Rotation: 75.30%, Translation: 91.40%, Scale: 88.80%).
We attribute the drop in the rotation prediction performance when using the Chamfer distance loss (68.40% meet the rotation threshold) compared to the Earth Mover's distance loss (75.30% meet the rotation threshold) to the one-to-one matching performed

in the Earth Mover's distance being more descriptive for errors in object pose.

Looking at our handcrafted baselines, we observe a good scale prediction performance. Most models learn to predict varying scales (positive scale predictions in 86.00% to 88.80% range) and outperform using a object-specific median scale (70.87%) for each prediction. The only exception being the Chamfer training on point clouds in full camera space (51.00%), which we mentioned earlier in this subsection.

We do not observe a good shape prediction performance. Our handcrafted shape baseline using the object-specific median-based shape (63.89% of the shape prediction are successful with a median IoU of 0.35) greatly outperforms all our proposed models (successful shape prediction fractions are in 46.30% to 48.00% range with median IoUs in 0.16 to 0.19 range). Our intuition is that the voxel decoder might not be expressive enough to generate good predictions under uncertainty, when the object is slightly occluded or obstructed by another object.

4.3. Scene Level Evaluation

In this experiment we compare the performance of *Factored3D* and our proposed network extensions when predicting a full scene (see subsection 4.2.2 for more details on our tested network extensions). The predicted scenes are transformed into scene voxel grids to allow for a unified numerical comparison of the scene level prediction performance.
We will first describe the transformation used to create a scene voxel grid based on the multiple factored object outputs in subsection 4.3.1 before we present our scene level evaluation results in subsection 4.3.2.

4.3.1. Scene Voxelization

A predicted scene from our tested networks consists of multiple factored object predictions in camera space. For this experiment we use the bounding box proposals for each input image instead of the ground-truth bounding boxes as in our evaluation in section 4.2. Using the bounding box proposals during inference means, that we do not have a definitive matching between a predicted object and its ground-truth factored representation. A proposal could be incorrect and not be close to any object in the ground-truth scene.
We transform the full scene into a voxel occupancy grid with $64 \times 32 \times 64$ voxels, where each voxel represents a $8cm \times 8cm \times 8cm$ volume. The objects are transformed into

world space. Many objects are axis-aligned in world space, which helps preserving the predicted object shape during the scene voxelization process. We store a corner point of the ground-truth voxel scene in world space as a reference to evaluate the same scene volume in the predicted scene in world space as well.
For this evaluation we do not distinguish between the different predicted object shapes, but only consider voxels in the scene as occupied or unoccupied by an object. A scene voxel is occupied if any object point is within the voxel volume.

4.3.2. Results

We evaluated four viewpoints from each house in the SUNCG test set, leading to around 16,000 images in total. In table 4.2, we evaluate the network extensions' predicted scene voxels based on their IoU with the ground-truth scene voxels. In general, the results of the scene level evaluation report similar findings as the results of the object level evaluation done in section 4.2.
The original network (*Factored3D* with a median IoU of 0.228) and the network using the Earth Mover's distance to fine-tune the rotation and shape factors (*Factored3D + pose-only* D_{EMD} with a median IoU of 0.223) perform best. The similar results between both methods do not indicate if the point cloud representation using the Earth Mover's distance can lead to substantial improvements over the original network architecture.

Similar as for the object level evaluation experiment, training *Factored3D* for one epoch only using the Chamfer distance measure also leads to poor results for the full scene prediction (*Solo* D_{CD} with a median IoU of 0.167).

In Figure 4.1, the full scene prediction performance of Factored3D + pose-only D_{EMD} and Factored3D are compared visually. Both networks perform well on simple scenes with not a lot of visual clutter (see row 1 and row 4 of Figure 4.1).
For cluttered scenes, both networks are susceptible to predicting too many objects in a scene. For instance in row 3 of Figure 4.1, both networks mistake the turquoise rugs for objects in the scene.

4.4. Results on Real Indoor Scenes

We also applied Factored3D and the network using the Earth Mover's distance to fine-tune the rotation and shape factors (*Factored3D + pose-only* D_{EMD}) on real indoor images from the NYU-Depth V2 data set [NF12]. The predictions are presented in

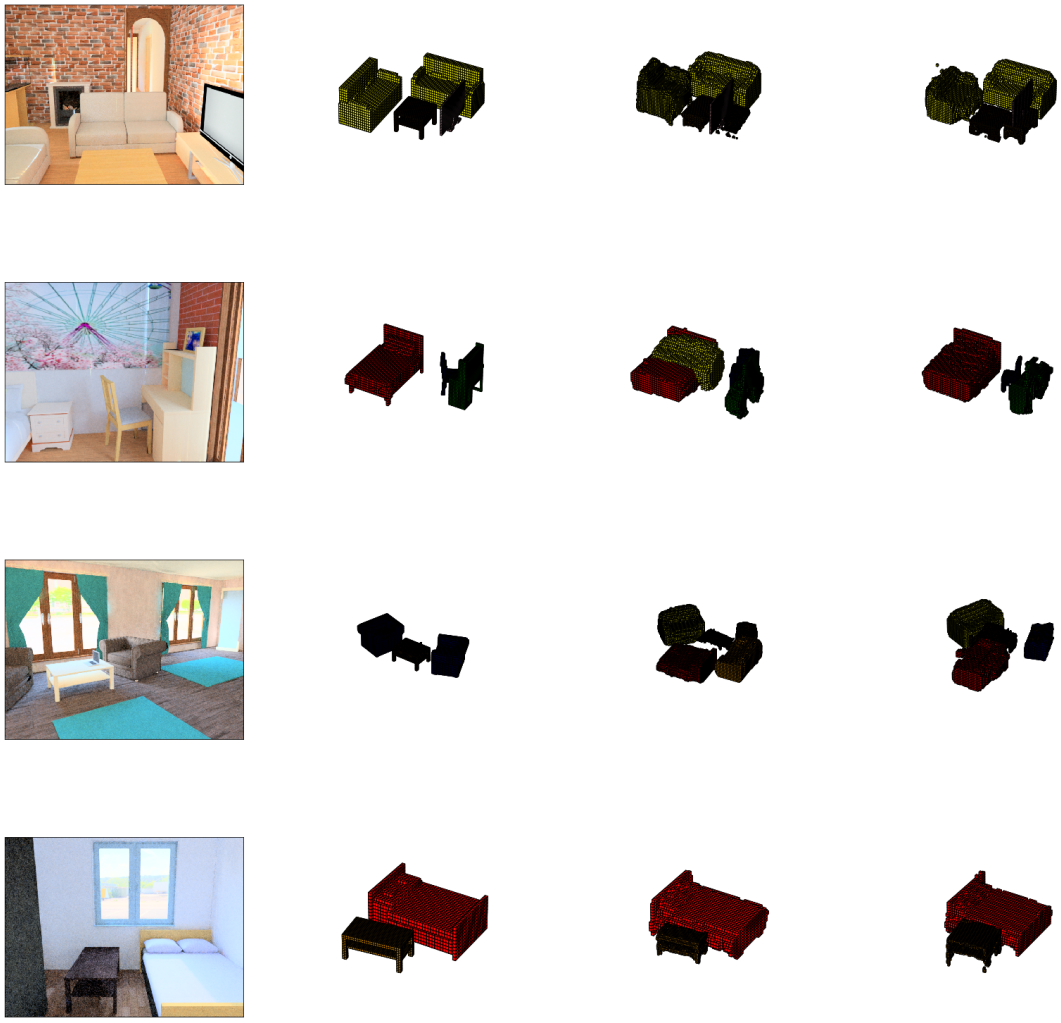


Figure 4.1.: Comparison of scene predictions of Factored3D + pose-only D_{EMD} and Factored3D on the SUNCG data set. Here we present the prediction of a scene containing all predicted objects for the given input image. The first and second column show the given input image from the SUNCG test set and the ground-truth object predictions of the scene. The third column shows the prediction of the Factored3D + pose-only D_{EMD} network extension. The fourth column shows the prediction of the Factored3D model.

method	%(IoU > 0.20)	Med-IoU
Factored3D	**56.52**	**0.228**
Factored3D + D_{CD}	53.48	0.215
Solo D_{CD}	41.35	0.167
Factored3D + pose-only D_{CD}	51.56	0.207
Factored3D + pose-only D_{EMD}	**55.54**	**0.223**

Table 4.2.: Scene prediction performance on input images from the SUNCG test set. In this table we compare the scene prediction performance of the different proposed network extensions presented in subsection 4.2.2. In this experiment are only considered images from the SUNCG test set that contain at least one object. We report the percentage of the predicted scenes that have a IoU larger than $\delta_{scene} = 0.20$ and the median IoU (Med-IoU) for each network.
The baseline network *Factored3D* and the network using the Earth Mover's distance to fine-tune the rotation and shape factors show the best performance for the full scene prediction task.

Figure 4.2.
The networks perform fairly well for scenes with a few objects, that do not overlap. For scenes with a lot of visual clutter, both networks perform poorly. Unfortunately, real world scenes tend to contain more objects than synthetic scenes. This makes training purely on synthetic data sets not ideal for networks that are then applied in the real world.

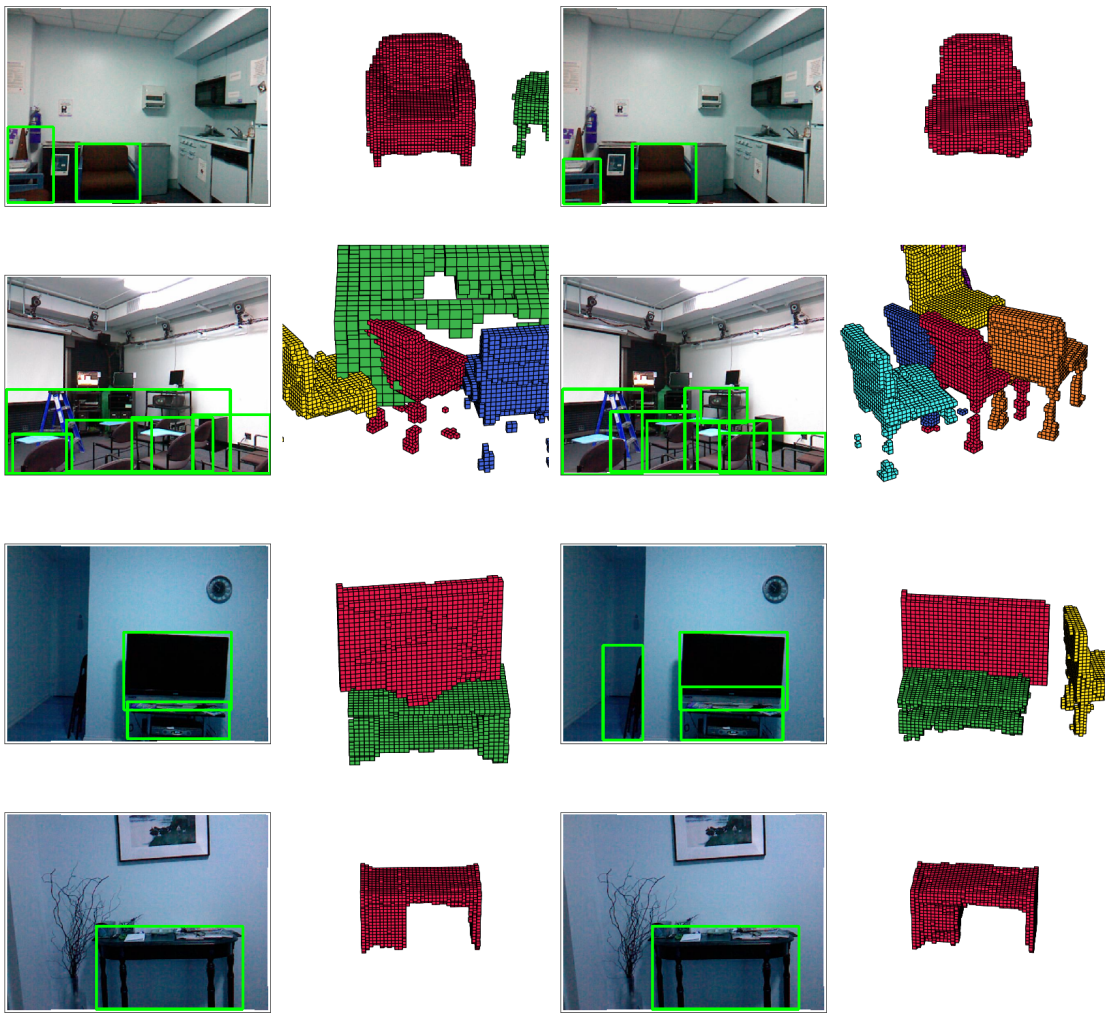


Figure 4.2.: Comparison of scene predictions of Factored3D + pose-only D_{EMD} and Factored3D on the NYU-Depth V2 data set. Here we present scene predictions made on real world indoor images. The first column shows the detected object bounding box of Factored3D + pose-only D_{EMD} and the second column shows the object prediction of Factored3D + pose-only D_{EMD}.
The third column shows the detected object bounding box of Factored3D and the fourth column shows the the object prediction of Factored3D.
Both networks perform well when the input scene does not contain many overlapping objects (see row 1, row 3, and row 4). For cluttered scenes both networks perform poorly and produce unreasonable scene predictions (see row 2).

5. Conclusions

In this chapter we will summarize the discoveries made based on the results presented in this work. We will also discuss the possibilities and limitations of using a factored object representation for 3D scene understanding tasks.

Retraining *Factored3D* on a different split of the SUNCG data set has confirmed the reported performance by Tulsiani et al. [Tul+18]. The shape prediction performance does not hold up to the reported results on the original *Factored3D*. Regardless, our mean shape comparison baseline suggests that the shape prediction only contains a very low level of detail. This might improve by replacing the voxel autoencoder with adversarial learning strategies similar to 3D-GAN [Wu+16] or the VAE/GAN approach by Larsen et al. [Lar+16].
For the remaining factors rotation, translation, and scale, *Factored3D* performs well and makes a good case, that factored representations can break up the complex task of 3D scene understanding from a single RGB image into more manageable subtasks for the network. Unfortunately, this approach requires data sets with full 3D supervision to obtain accurate ground-truth factors, which at the moment limits training to synthetic data sets.

Our point cloud transformation procedure presents one possible implementation to convert a factored representation to a 3D point cloud representation. We limited our experimentation on this representation to point cloud based loss functions. It is possible to extend the network with layers from PointNet [Qi+17a; Qi+17b] and FoldingNet [Yan+18c], which were tailored for point clouds. This would also make the already complex *Factored3D* training procedure even harder to reproduce.
Some combinations of certain factors and point cloud loss functions (e.g.: *pose-only* D_{EMD}) have shown promising results, but cannot surpass the performance of the original factored training approach on *Factored3D*. More experimentation should provide better indications which particular factors could benefit from losses calculated on a point cloud representation.

Overall, we believe that factored representations, with their human-readable representation of the objects in the scene, can be more easily extended to use geometric and

physical reasoning [Zhe+13] or domain-specific scene statistics [CY01] to fine-tune their scene predictions for realistic prediction results.

Glossary

6D pose estimation 6D pose estimation refers "to estimating the rigid transformation from the object coordinate system O to the camera coordinate system C" [Xia+18]. This can be described using a 3D rotation and a 3D translation, leading to a total of 6 degrees of freedom. 7

amodal We use the term amodal in the context of amodal perception of a scene. This refers to perceiving the scene layout while ignoring the objects in the scene. The scene layout only consists of the floor, walls, and the ceiling of an indoor scene. 7–10

anisotropic This describes the property of being directionally dependent. We use this term to describe the scaling factor, which is non-uniform among the x, y, and z axes. 23

dilated convolutional layers Dilated convolutional layers are often used for dense prediction, because of their exponential growth of the receptive field compared to the linear growth of the receptive field for regular convolutional layers [YK16]. This is achieved by constraining the kernel only on a subset of the actual input based on a dilation factor. 4

Factored3D This is how we refer to the factored prediction network proposed by Tulsiani et al. [Tul+18]. While the network was not officially named this way in the original paper, follow-up research of the same authors refers to it as *Factored3D* and we follow this convention [Kul+]. 1, 7–9, 11, 13–16, 18, 19, 21–24, 27–36, 44

IoU We use the Intersection over Union (IoU) metric to measure the fraction of overlap between two areas in 2D and the fraction of volume overlap between two volumes in 3D. It is also known as the Jaccard index. Given the two sets S_1 and S_2, we define it as: $\text{IoU}(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$. 12, 16, 28, 31, 32, 34

medoid A medoid is a data point within a data set that has a minimal average dissimilarity to all other members in the data set. In the context of k-medoids clustering, the k cluster centers are represented by the medoid of the cluster. Tulsiani et al. used k-medoids clustering to produce the 24 quaternion bins for the rotation classification task [Tul+18]. 21

physically-based rendering The approach to render graphics under consideration of photogrammetry and using accurate models for the flow of light is summed up under the term physically-based rendering. The goal is to create photorealistic images. Mitsuba is a rendering system in the style of physically-based rendering [Jak]. 19, 21

ROI-Pooling This a specific pooling layer that produces a fixed output feature map from a non-uniform input size. The input is referred to as the Region-of-Interest (ROI). The ROI-pooling layer divides the input into equal-sized sections and applies the pooling operation on each of these sections, which leads to the same output size regardless of the given input size. 9–11

Acronyms

ANN Artificial Neural Network. 1, 4, 21

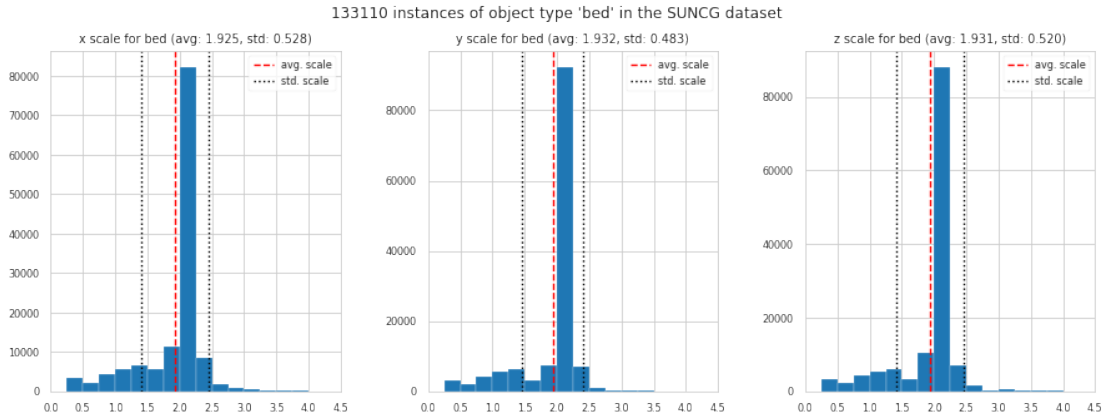
CNN Convolutional Neural Network. 3

IoU Intersection over Union. 38

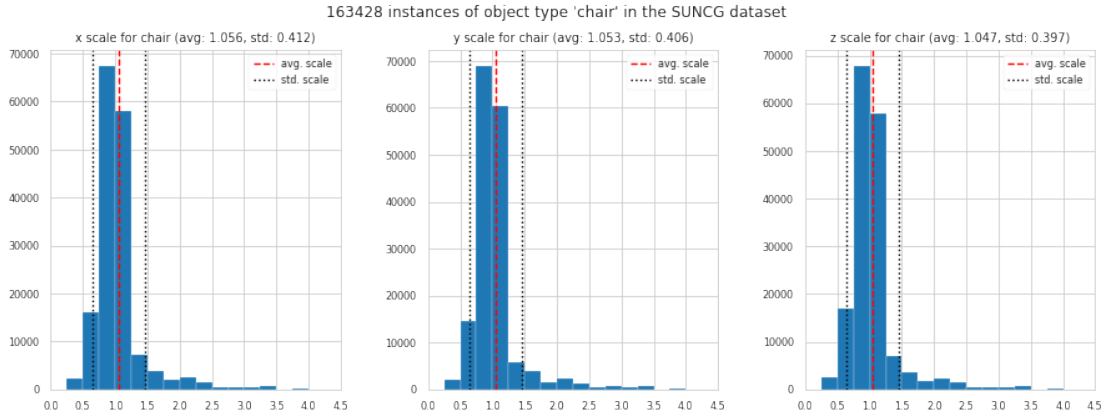
RGB **R**ed, **G**reen, and **B**lue color channels. 1, 2, 8, 9, 11, 17, 27, 36

ROI Region-of-Interest. 39

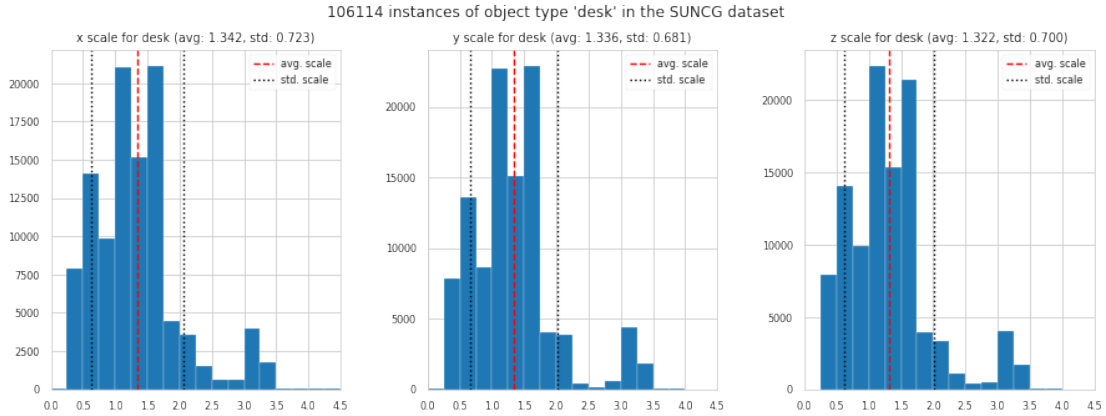
A. Appendix



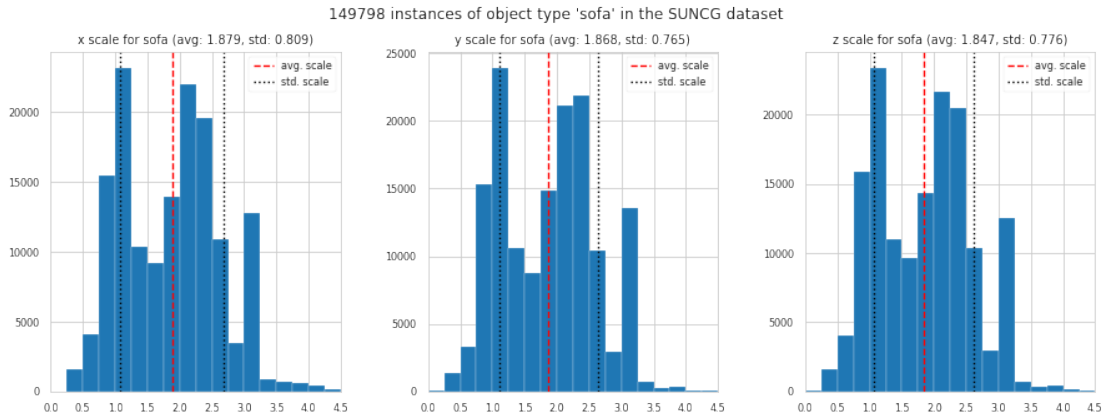
(a) Scale distribution of all objects of category bed in the SUNCG data set.



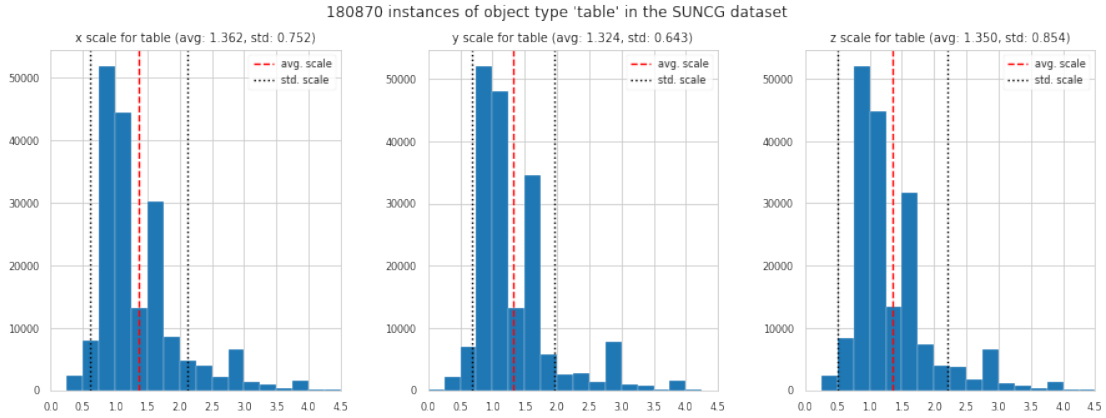
(b) Scale distribution of all objects of category chair in the SUNCG data set.



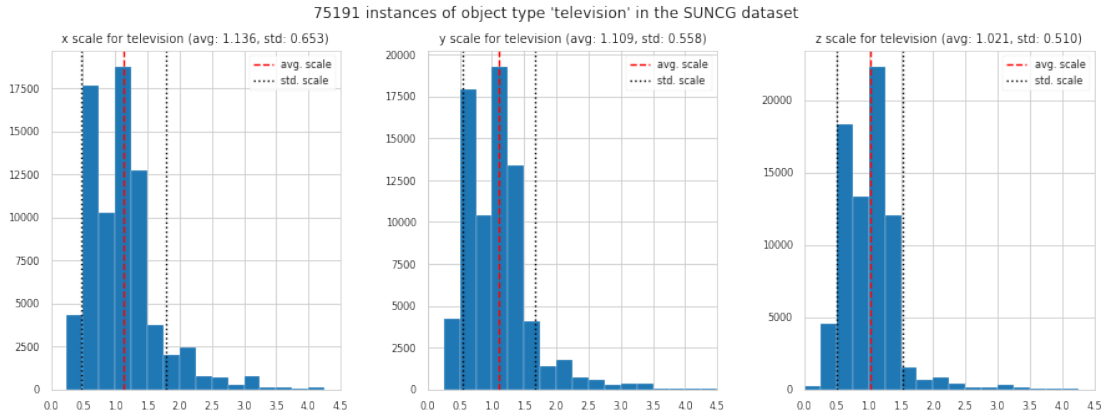
(c) Scale distribution of all objects of category desk in the SUNCG data set.



(d) Scale distribution of all objects of category sofa in the SUNCG data set.



(e) Scale distribution of all objects of category table in the SUNCG data set.



(f) Scale distribution of all objects of category television in the SUNCG data set.

Figure A.1.: Distribution of the Object Scales in the SUNCG Data Set. These histograms show the distribution of the object scale among the six object categories trained for prediction. The objects of category **bed** and **chair** show only very small variance in each scale dimension. This observation motivated the creation of a comparison baseline based on the category-specific median scales, since the scale prediction on objects of type **bed** and **chair** look easy based on the histograms. The objects of category **desk**, **sofa**, and **table** show larger variances in scale.

List of Figures

List of Tables

Bibliography

[Ach+18] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. "Learning Representations and Generative Models for 3D Point Clouds." In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. July 2018, pp. 40–49.

[Bar+77] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. "Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching." In: *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI'77. 1977, pp. 659–663.

[Bay+08] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. "Speeded-Up Robust Features (SURF)." In: *Comput. Vis. Image Underst.* 110.3 (June 2008), pp. 346–359.

[Ber85] D. P. Bertsekas. "A distributed asynchronous relaxation algorithm for the assignment problem." In: *1985 24th IEEE Conference on Decision and Control*. Dec. 1985, pp. 1703–1704.

[BT73] T. O. Binford and J. M. Tenenbaum. "Computer vision." In: *Computer* 6.5 (May 1973), pp. 19–24.

[Cab+08] S. Cabello, P. Giannopoulos, C. Knauer, and G. Rote. "Matching point sets with respect to the Earth Mover's Distance." In: *Computational Geometry* 39.2 (2008), pp. 118–133.

[Cha+15] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep. arXiv:1512.03012 [cs.GR]. Stanford University, Princeton University, Toyota Technological Institute at Chicago, 2015.

[Cha+17] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. "Matterport3D: Learning from RGB-D Data in Indoor Environments." In: *International Conference on 3D Vision (3DV)* (2017).

[Che+16] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. "Monocular 3D Object Detection for Autonomous Driving." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 2147–2156.

[Cho+16] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction." In: *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII*. 2016, pp. 628–644.

[CY01] J. M. Coughlan and A. L. Yuille. "The Manhattan World Assumption: Regularities in Scene Statistics which Enable Bayesian Inference." In: *Advances in Neural Information Processing Systems 13*. 2001, pp. 845–851.

[Den+09] J. Deng, W. Dong, R. Socher, L.-j. Li, K. Li, and L. Fei-fei. "Imagenet: A large-scale hierarchical image database." In: *In CVPR*. 2009.

[Dil] C. Diller. *Chamfer Distance for pyTorch*. URL: `https://github.com/chrdiller/pyTorchChamferDistance` (visited on 05/13/2019).

[DT05] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection." In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. June 2005, pp. 886–893.

[Ede] M. Eder. *PyTorch EMDLoss*. URL: `https://github.com/meder411/PyTorch-EMDLoss` (visited on 05/13/2019).

[EW11] S. M. Eslami and C. Williams. "Factored Shapes and Appearances for Parts-based Object Understanding." In: *BMVC*. 2011.

[Fir+16] M. Firman, O. M. Aodha, S. Julier, and G. J. Brostow. "Structured Completion of Unobserved Voxels from a Single Depth Image." In: *Computer Vision and Pattern Recognition (CVPR)*. 2016.

[Fir16] M. Firman. "RGBD Datasets: Past, Present and Future." In: *CVPR Workshop on Large Scale 3D Data: Acquisition, Modelling and Analysis*. 2016.

[FSG17] H. Fan, H. Su, and L. J. Guibas. "A Point Set Generation Network for 3D Object Reconstruction from a Single Image." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 2463–2471.

[GEH10] A. Gupta, A. A. Efros, and M. Hebert. "Blocks World Revisited: Image Understanding Using Qualitative Geometry and Mechanics." In: *European Conference on Computer Vision(ECCV)*. 2010.

[Gei+] A. Geiger, P. V. Gehler, V. Jampani, and C. Wang. *Scene Understanding*. URL: `https://ps.is.tuebingen.mpg.de/research_projects/scene-understanding` (visited on 03/17/2019).

[Gir+16] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta. "Learning a Predictable and Generative Vector Representation for Objects." In: *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI*. 2016, pp. 484–499.

[Gir15] R. Girshick. "Fast R-CNN." In: *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*. ICCV '15. 2015, pp. 1440–1448.

[Goo+14] 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, pp. 2672–2680.

[Gro+18] T. Groueix, M. Fisher, V. G. Kim, B. Russell, and M. Aubry. "AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation." In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[He+16] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition." In: *CVPR*. IEEE Computer Society, 2016, pp. 770–778.

[HSF18] G. Hinton, S. Sabour, and N. Frosst. "Matrix capsules with EM routing." In: 2018.

[Ink] N. Inkawhich. *Finetuning Trochvision Models*. URL: `https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html` (visited on 04/13/2019).

[Jak] W. Jakob. *Mitsuba physically based renderer*. URL: `http://www.mitsuba-renderer.org/` (visited on 04/25/2019).

[KAP13] J. Kober, J. Andrew Bagnell, and J. Peters. "Reinforcement Learning in Robotics: A Survey." In: *The International Journal of Robotics Research* 32 (Sept. 2013), pp. 1238–1274.

[Kul+] N. Kulkarni, I. Misra, S. Tulsiani, and A. Gupta. *3D-RelNet: Joint Object and Relational Network for 3D Prediction*. URL: `http://www.cs.cmu.edu/~nileshk/papers/3drelnet.pdf` (visited on 05/12/2019).

[Kul+15] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. "Deep Convolutional Inverse Graphics Network." In: *Advances in Neural Information Processing Systems 28*. 2015, pp. 2539–2547.

[Lar+16] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. "Autoencoding Beyond Pixels Using a Learned Similarity Metric." In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. 2016, pp. 1558–1566.

[Low99] D. G. Lowe. "Object Recognition from Local Scale-Invariant Features." In: *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*. ICCV '99. 1999, pp. 1150–. ISBN: 0-7695-0164-8.

[LVV18] L. Lettry, K. Vanhoey, and L. Van Gool. "Unsupervised Deep Single-Image Intrinsic Decomposition using Illumination-Varying Image Sequences." In: *Computer Graphics Forum* 37 (Oct. 2018), pp. 409–419.

[Mum91] D. Mumford. "Mathematical theories of shape: do they model perception?" In: *Geometric Methods in Computer Vision* 1570 (Sept. 1991).

[NF12] P. K. Nathan Silberman Derek Hoiem and R. Fergus. "Indoor Segmentation and Support Inference from RGBD Images." In: *ECCV*. 2012.

[NKP18] M. Naseer, S. H. Khan, and F. Porikli. "Indoor Scene Understanding in 2.5/3D: A Survey." In: *CoRR* (2018).

[NMa+16] N.Mayer, E.Ilg, P.Häusser, P.Fischer, D.Cremers, A.Dosovitskiy, and T.Brox. "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation." In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[Pas+17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. "Automatic differentiation in PyTorch." In: *NIPS-W*. 2017.

[Pla] Planner5D. *Planner5D home design tool*. URL: `https://planner5d.com/` (visited on 03/18/2019).

[Qi+17a] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation." In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* (2017).

[Qi+17b] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space." In: *arXiv preprint arXiv:1706.02413* (2017).

[Ren+15] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks." In: *Proceedings of the 28th International Conference on Neural Information Processing*. NIPS'15. 2015, pp. 91–99.

[RPB15] O. Ronneberger, P.Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Vol. 9351. 2015, pp. 234–241.

[RUG17] G. Riegler, O. Ulusoy, and A. Geiger. "OctNet: Learning Deep 3D Representations at High Resolutions." In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017*. Piscataway, NJ, USA: IEEE, July 2017.

[SFH17] S. Sabour, N. Frosst, and G. E. Hinton. "Dynamic Routing Between Capsules." In: *Advances in Neural Information Processing Systems 30*. 2017, pp. 3856–3866.

[SFH18] D. Shin, C. Fowlkes, and D. Hoiem. "Pixels, voxels, and views: A study of shape representations for single view 3D object shape prediction." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[SNS09] M. Sun, A. Y. Ng, and A. Saxena. "Make3D: Learning 3D Scene Structure from a Single Still Image." In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 31.05 (May 2009), pp. 824–840.

[Son+17] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. "Semantic Scene Completion from a Single Depth Image." In: *Proceedings of 29th IEEE Conference on Computer Vision and Pattern Recognition* (2017).

[Su+15] H. Su, C. R. Qi, Y. Li, and L. J. Guibas. "Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views." In: *The IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[SZ14] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." In: *CoRR* (2014).

[TDB17] M. Tatarchenko, A. Dosovitskiy, and T. Brox. "Octree Generating Networks: Efficient Convolutional Architectures for High-Resolution 3D Outputs." In: *The IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[TM15] S. Tulsiani and J. Malik. "Viewpoints and keypoints." In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 1510–1519.

[Tul+] S. Tulsiani, S. Gupta, D. Fouhey, A. A. Efros, and J. Malik. *Code release for "Factoring Shape, Pose, and Layout from the 2D Image of a 3D Scene"*. URL: `https://github.com/shubhtuls/factored3d` (visited on 04/24/2019).

[Tul+17] S. Tulsiani, H. Su, L. J. Guibas, A. A. Efros, and J. Malik. "Learning Shape Abstractions by Assembling Volumetric Primitives." In: *Computer Vision and Pattern Regognition (CVPR)*. 2017.

[Tul+18] S. Tulsiani, S. Gupta, D. Fouhey, A. A. Efros, and J. Malik. "Factoring Shape, Pose, and Layout from the 2D Image of a 3D Scene." In: *Computer Vision and Pattern Recognition (CVPR)*. 2018.

[Wan+17] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong. "O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis." In: *ACM Transactions on Graphics (SIGGRAPH)* 36.4 (2017).

[Wan+18] Y. Wang, D. J. Tan, N. Navab, and F. Tombari. "Adversarial Semantic Scene Completion from a Single Depth Image." In: *Sixth International Conference on 3D Vision, 3DV 2018*. Sept. 2018, pp. 426–434.

[Wu+] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. *The Princeton ModelNet*. URL: `http://modelnet.cs.princeton.edu/` (visited on 04/06/2019).

[Wu+15] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. "3D ShapeNets: A deep representation for volumetric shapes." In: *CVPR*. IEEE Computer Society, 2015, pp. 1912–1920. ISBN: 978-1-4673-6964-0.

[Wu+16] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. "Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling." In: *Advances in Neural Information Processing Systems 29*. 2016, pp. 82–90.

[Xia+18] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes." In: 2018.

[Yan+18a] B. Yang, Z. Lai, X. Lu, S. Lin, H. Wen, A. Markham, and N. Trigoni. "Learning 3D Scene Semantics and Structure From a Single Depth Image." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2018.

[Yan+18b] B. Yang, S. Rosa, A. Markham, N. Trigoni, and H. Wen. "Dense 3D Object Reconstruction from a Single Depth View." In: *TPAMI*. 2018.

[Yan+18c] Y. Yang, C. Feng, Y. Shen, and D. Tian. "FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation." In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, pp. 206–215.

[YK16] F. Yu and V. Koltun. "Multi-Scale Context Aggregation by Dilated Convolutions." In: *ICLR*. 2016.

[Yua+18] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert. "PCN: Point Completion Network." In: *3D Vision (3DV), 2018 International Conference on*. 2018.

[Zam+18] M. Zamorski, M. Zięba, R. Nowak, W. Stokowiec, and T. Trzciński. "Adversarial Autoencoders for Generating 3D Point Clouds." In: *arXiv e-prints* (Nov. 2018).

[ZD14] C. L. Zitnick and P. Dollár. "Edge Boxes: Locating Object Proposals from Edges." In: *ECCV*. 2014.

[Zha+17] Y. Zhang, S. Song, E. Yumer, M. Savva, J.-Y. Lee, H. Jin, and T. Funkhouser. "Physically-Based Rendering for Indoor Scene Understanding Using Convolutional Neural Networks." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[Zha+18] Y. Zhao, T. Birdal, H. Deng, and F. Tombari. "3D Point-Capsule Networks." In: *CoRR* abs/1812.10775 (2018).

[Zhe+13] B. Zheng, Y. Zhao, J. C. Yu, K. Ikeuchi, and S. Zhu. "Beyond Point Clouds: Scene Understanding by Reasoning Geometry and Physics." In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. June 2013, pp. 3127–3134.