Factoring Shape, Pose, and Layout from the 2D Image of a 3D Scene

Shubham Tulsiani, Saurabh Gupta, David Fouhey, Alexei A. Efros, Jitendra Malik
University of California, Berkeley
{shubhtuls, sgupta, dfouhey, efros, malik}@eecs.berkeley.edu

Abstract

The goal of this paper is to take a single 2D image of a scene and recover the 3D structure in terms of a small set of factors: a layout representing the enclosing surfaces as well as a set of objects represented in terms of shape and pose. We propose a convolutional neural network-based approach to predict this representation and benchmark it on a large dataset of indoor scenes. Our experiments evaluate a number of practical design questions, demonstrate that we can infer this representation, and quantitatively and qualitatively demonstrate its merits compared to alternate representations.

1. Introduction

How should we represent the 3D structure of the scene in Figure 1? Most current methods for 3D scene understanding produce one of two representations: i) a 2.5D image of the scene such as depth [7, 26] or surface normals [4, 9]; or ii) a volumetric occupancy grid/voxels representation in terms of a single voxel grid [5, 10, 33]. Accordingly, they miss a great deal. First, all of these representations erase distinctions between objects and would represent Figure 1 as an undifferentiated soup of surfaces or volumes rather than a set of chairs next to a table. Moreover, the 2.5D representations intrinsically cannot say anything about the invisible portions of scenes such as the thickness of a table or presence of chair legs. While in principle voxel-based representations can answer these questions, they mix together beliefs about shape and pose and cannot account for the fact that it is easy to see that the chair has a thin back but difficult to determine its exact depth.

We present an alternative: we should think of scenes as being composed of a distinct set of factors. One represents the *layout*, which we define as the scene surfaces that enclose the viewer, such as the walls and floor, represented in terms of their full, amodal extent *i.e.* what the scene would look like without the objects. The others represent a discrete set of objects which are in turn factored into *3D shape* (voxels) and *pose* (location and rotation). This representation solves a number of key problems: rather than a muddled mess of voxels, the scene is organized into discrete entities, permitting subsequent tasks to reason in terms of questions like "what would the scene be like if I moved that chair." In terms of reconstruction itself, the factored approach does not conflate uncertainties in pose and shape, and automatically allocates voxel resolution, enabling high resolution output for free.

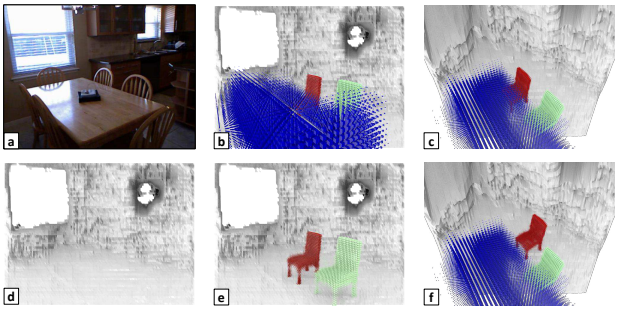

Figure 1: Our 3D representation. Given a single 2D image (a) we infer a 3D representation which is *factored* in terms of a set of objects inside an enclosed volume. We show it from the camera view in (b) and a novel view in (c). By virtue of being factored, our representation trivially enables answering questions that are impossible in other ones. For example, we can give the scene (d) without any objects; (e) without the table; or (f) answer "what would it be like if I moved the chair". These and all results best viewed in color on screen.

One needs a way to infer this representation from single 2D images. We thus propose an approach in Section 3 which is summarized in Figure 2. Starting with an image and generic object proposals, we use convolutional neural networks (CNNs) to predict both the layout, *i.e.* amodal scene surfaces, as well as the underlying shape and pose of objects. We train this method using synthetic data [30], although we show it on both synthetic and natural data.

We investigate a number of aspects of our method in Section 4. Since our approach is the first method to tackle this task and many design decisions are non-obvious, we first present extensive ablations in Section 4.3. We then demonstrate that we can infer the full representation and find current performance limitations in Section 4.4. Next, since one

Project website with code: https://shubhtuls.github.io/factored3d/

might naturally wonder how the representation compares to others, we compare it to the more standard representations of a per-pixel depthmap and single voxel grid in Section 4.5. Figure 6 qualitatively and Figure 8 quantitatively demonstrate the benefits of our representation. We finally show qualitative results on the NYUv2 dataset.

2. Related Work

Our work aims to take a single 2D image and factor it into a set of constituent 3D components, and thus touches on a number of topics in 3D scene understanding.

The goal of recovering 3D properties from a 2D image has a rich history in computer vision starting from Robert's Blocks World [24]. In the learning-based era, this has mainly taken the form of estimating view-based per-pixel 3D properties of scenes such as depth [7, 26] or orientation information [9, 16]. These approaches are limited in the sense that they intrinsically cannot say anything about non-visible parts of the scene. This shortcoming has motivated a line of work aiming to infer volumetric reconstructions from single images [5, 10, 33], working primarily with voxels. These have been exclusively demonstrated with presegmented objects in isolation, and never with scenes: scenes pose the additional challenges of delineating objects, properly handling uncertainty in shape and pose, and scaling up resolution. Our representation automatically and naturally handles each of these challenges.

Our goal of a volumetric reconstruction of a scene has been tackled under relaxed assumptions that alleviate or eliminate the difficulty of handling either shape or pose. For example, with RGBD input, one can complete the invisible voxels from the visible ones as in [30]. Here, the problem of pose is eliminated: because of the depth sensor, one knows where the objects are, and the remaining challenge is inferring the missing shape. Similarly, in CAD retrieval scenarios, one assumes the object [3, 4, 13, 19, 20] or scene [17] can be represented in terms of a pre-determined dictionary of shapes; a great deal of earlier work [12, 18, 27] tackled this with a dictionary of box models. The challenge then is to detect these objects and figure out their pose. In contrast, we jointly infer both shape and pose. Our approach, therefore does not rely on privileged information such as the precise location of the visible pixels or is restricted to a set of pre-determined objects.

In the process of predicting our representation, we turn to tools from the object detection literature. There is, of course, a large body of work between classic 2D detection and full 3D reconstruction. For instance, researchers have predicted 3D object pose [23, 32], low-dimensional parametric shapes [8, 34], and surface normals [28]. Our representation is richer than this past work, providing detailed volumetric shape and pose, as well as the layout of the rest of the scene.

3. Approach

The goal of our method is to take a single 2D image and infer its 3D properties in terms of scene layout, object shape and pose. We attack this problem with two main components, illustrated in Figure 2, that can be trained end-to-end. The first is a scene network that maps a full image to an amodal layout describing the scene minus the object. The second is an object-centric network that maps bounding boxes to the their constituent factors: shape and pose. We now describe the architectural details of each component, the loss functions learned to learn each, and the training and inference procedures. We present a sketch of the architecture in favor of spending more time on presenting experiments; we follow standard design decisions but full details appear in the supplemental [1].

3.1. Layout

We first predict the *layout*. This represents the enclosing surfaces of the scene, such as the walls and floor. Specifically, the layout is the amodal extent of these surfaces (*i.e.* the floor as it exists behind the objects of the scene). Past approaches to this [15, 27] have primarily posed this in terms of fitting a vanishing-point-aligned 3D box, which intrinsically cannot generalize to non-box environments. Here, we treat the more general case as 2.5D problem and propose to predict the layout as the disparity (*i.e.* inverse depth) map of the scene as if there were no objects.

We predict the layout using the layout module, a skip-connected network similar to [21]. The first half of the network takes the image and maps it to an intermediate representation, slowly decreasing spatial resolution and increasing increasing feature channel count. The latter half, upconvolves in the reverse fashion while concatenating features from the encoder. We train our network end-to-end using the L_1 objective, or if $\hat{\mathbf{H}}$ denotes our prediction and $\mathbf{H}$ the ground-truth layout, $\mathrm{L}_H = ||\mathbf{H} - \hat{\mathbf{H}}||_1$.

3.2. Object Predictions

We represent the shape of an object as a 32^3 voxel occupancy grid $\mathbf{V}$ and the pose as anisotropic scaling $\mathbf{c}$, followed by a rotation represented by a quaternion $\mathbf{q}$, and finally a translation $\mathbf{t}$. Without any other constraints, this representation is underdetermined: one can apply many types of changes to the shape and undo them in the pose. Therefore, we represent the shape in a canonical (*i.e.* front-facing and upright) coordinate frame which is normalized and centered so the object dimensions vary between $[-0.5\mathrm{m}, 0.5\mathrm{m}]$. This in turn specifies the pose.

Architecture and features. First, we describe how the system maps an image and bounding box to this representation; the following subsection explains how this is done for a set of regions. Given a feature vector, linear layers map directly

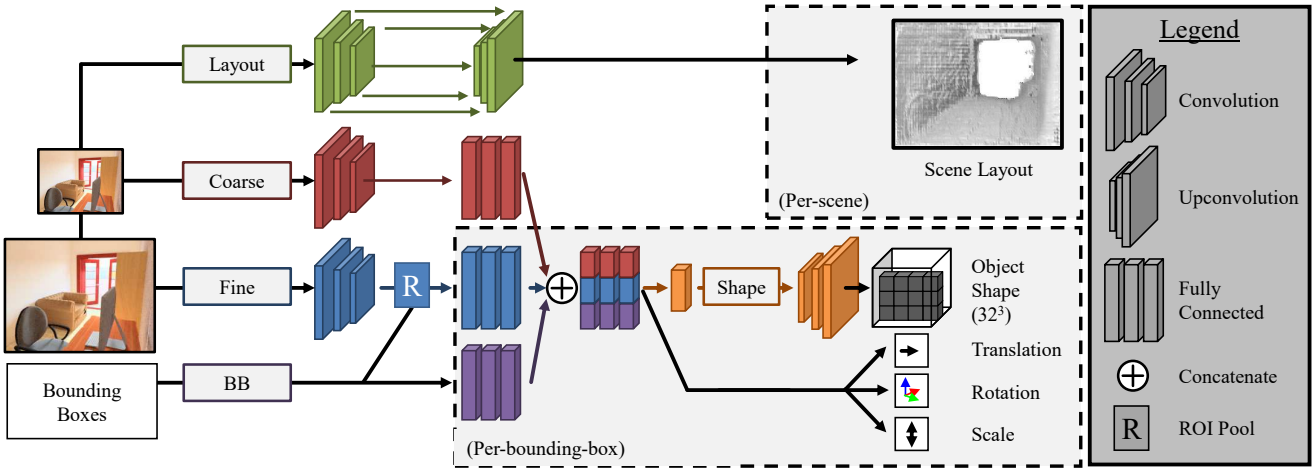

Figure 2: Overview of our framework. We take as input an image and set of bounding boxes. The scene layout $\mathbf{H}$ is predicted by the layout module, a skip-connected CNN. Each bounding box is then represented by features from three sources: ROI-pooled features extracted from a fine module that uses the original resolution, full image context features from the coarse module, and the bounding-box location. These features are concatenated and passed through several layers, culminating in the prediction of a shape code $\mathbf{s}$, scale $\mathbf{c}$, translation $\mathbf{t}$, and rotation of the object $\mathbf{q}$. The shape code is mapped to voxels $\mathbf{V}$ by the shape decoder.

to $\mathbf{t}$, $\mathbf{q}$, and $\mathbf{c}$. Since $\mathbf{V}$ is high dimensional and structured, we first map to a shape code $\mathbf{s}$ which is then reshaped and upconvolved to $\mathbf{V}$.

We use three sources to construct our feature vector. The primary one is the fine module, which maps the image at its original resolution to convolutional feature maps, followed by ROI pooling to obtain features for the window [11]. As additional information, we also include fixed-length features from: (1) a coarse module that maps the entire image at a lower resolution through convolutional layers then vectorizes it; and (2) bounding box module mapping the bounding box location through fully connected layers. The three sources are concatenated. In the experiment section, we report experiments without the contextual features.

Shape loss. The shape of the object is a discrete volumetric grid $\mathbf{V}$, which we decode from a fixed-length shape code $\mathbf{s}$ using the shape decoder. Our final objective is a per-voxel cross-entropy loss between the prediction $\hat{\mathbf{V}}$ and ground-truth $\mathbf{V}$, or

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

In practice, this objective is difficult to optimize, so we bootstrap the network following [10]. We learn an autoencoder on voxels whose bottleneck and decoder match our network in size. In the first stage, the network learns to mimic the autoencoder: given a window of the object, we minimize the L_2 distance between the autoencoder's bottleneck representation and the predicted shape code. The voxel decoder is then initialized with the autoencoder's decoder and the network is optimized jointly to minimize L_V.

Rotation Prediction. We parameterize rotation with a unit-normalized quaternion. We found that framing the problem as classification as in [31,32] handled the multi-modality of the problem better. We cluster the quaternions in the training set into 24 bins and predict a probability distribution $\mathbf{k}_d$ over them. Assuming k denotes the ground-truth bin, we minimize the negative log-likelihood,

$$\mathrm{L}_q^c = -\log(\mathbf{k}_d^k). \quad (2)$$

The final prediction is then the most likely bin. We evaluate the impact of this choice in the experiment section and compare it with a standard squared Euclidean loss.

Scale and translation prediction. Finally, anisotropic scaling and translation are formulated as regression tasks, and we minimize the squared Euclidean loss (in log-space for scaling):

$$\mathrm{L}_t = ||\mathbf{t}-\hat{\mathbf{t}}||_2^2; \quad \text{and} \quad \mathrm{L}_c = ||\log(\mathbf{c}) - \log(\hat{\mathbf{c}})||_2^2. \quad (3)$$

3.3. Training to Predict A Full Scene

We now describe how to put these components together to predict the representation for a full scene: so far, we have described how to predict with the boxes given as opposed to the case where we do not know the boxes a-priori. At training time, we assume that we have a dataset of annotated images in which we have the box as well as corresponding 3D structure information (i.e., pose, shape, etc.).

Proposals. To handle boxes, we use an external bounding-box proposal source and predict, from the same features as those used for object prediction, a foreground probability f representing the probability that a proposal corresponds to

a foreground object and optimize this with a cross-entropy loss. If $\mathcal{B}^+$ and $\mathcal{B}^-$ represent foreground and background proposals, our final objective is

$$\sum_{b \in \mathcal{B}^+} (\mathrm{L}_V + \mathrm{L}_q + \mathrm{L}_t + \mathrm{L}_r - \ln(f)) + \sum_{b \in \mathcal{B}^-} \ln(1-f),$$

which discriminates between foreground and background proposals and predicts the 3D structure corresponding to foreground proposals.

For proposals, we use 1K edge boxes [37] proposals per image. We assign proposals to ground truth objects based on modal 2D bounding box IoU. We treat proposals with more than 0.7 IoU as foreground-boxes ($\mathcal{B}^+$) and those with less than 0.3 IoU with any ground truth object as background boxes ($\mathcal{B}^-$).

Training Details. We initialize the coarse and fine convolution modules with Renset-18 [14] pretrained on ILSVRC [25] and all other modules randomly. We train the object network for 8 epochs with the autoencoder mimicking loss and then 1 additional epoch with the volumetric loss. Full model specification appears in the supplemental [1].

4. Experiments

We now describe the experiments done to validate our proposed representation and the described approach for predicting it. We first introduce the datasets that we use, one synthetic and one real, and the metrics used to evaluate our rich representation. Since our approach is the first to predict this representation, we begin by analyzing a number of design decisions by evaluating shape prediction in isolation. We then analyze the extent to which we can predict the representation and identify current performance bottlenecks. Having demonstrated that we can infer our representation, we compare the representation itself with alternate ones both qualitatively and quantitatively. Finally, we show some results on natural images.

4.1. Datasets

We use two datasets. The first is SUNCG, introduced by Song *et al*. [30]. The dataset consists of 3D models of houses created by users on an online modeling platform and has a diverse set of scenes with numerous objects and realistic clutter and therefore provides a challenging setup to test our approach. We use the physically-based renderings provided by Zhang *et al*. [35] for our experiments and randomly partition the houses into a 70%-10%-20% train, validation and test split. Overall, we obtain over 400,000 rendered training images and for each image we associate the visible objects with their corresponding 3D code by parsing the available house model. However, the objects present in the images are often too diverse *e.g*. fruit-baskets, ceiling lights, doors, candlesticks *etc*. and detecting and reconstructing these is extremely challenging so we restrict the set of ground-truth boxes to only correspond to a small but diverse set of indoor object classes – bed, chair, desk, sofa, table, television. The second is NYU [29], which we use to verify qualitatively that our model is able to generalize, without additional training, to natural images.

4.2. Metrics

Our method and representation subsumes a number of different past works and thus there is no standard way of evaluating it. We therefore break down the components of our approach and use the standard evaluation metrics for each component (*i.e*. $\mathbf{V}$, $\mathbf{q}$, $\mathbf{t}$, and $\mathbf{c}$). For each component, we define an error Δ that measures the discrepancy between the predicted value and the ground truth as well as a threshold δ that defines a true positive in the detection setting. For evaluating shape prediction in isolation, we aggregate results by taking the median over Δ and fraction of instances with distance below (or for IoU, overlap above) δ.

Shape ($\mathbf{V}$): We use the standard [5] protocol and set Δ_V to measuring intersection over union (IoU) and use as threshold $\delta_V = 0.25$.

Rotation ($\mathbf{q}$): We compute the geodesic distance between two rotations, or $\Delta_q(\mathbf{R}_1, \mathbf{R}_2) = (2)^{-1/2} || \log(\mathbf{R}_1^T \mathbf{R}_2) ||_F$. We set $\delta_q = \frac{\pi}{6}$ following [32].

Scale ($\mathbf{c}$): We define distance as the average logarithmic difference in scaling factors, or $\Delta(\mathbf{c}_1, \mathbf{c}_2) = \frac{1}{3} \sum_{i=1}^{3} |\log_2(\mathbf{c}_1^i) - \log_2(\mathbf{c}_2^i)|$. We threshold at $\delta_c = 0.5$, corresponding to being within a factor of $\sqrt{2}$.

Translation ($\mathbf{t}$): We use the standard Euclidean distance $\Delta_t(\mathbf{t}_1, \mathbf{t}_2) = ||\mathbf{t}_1 - \mathbf{t}_2||$ and threshold at $\delta_t = 1$m.

2D Bounding Box ($\mathbf{b}$): In the detection setting, we also consider the 2D bounding box ($\mathbf{b}$) and define Δ_b and δ_B as standard 2D IoU with the standard threshold of 0.5.

Detection Metrics. In the detection setting, we combine these metrics and define a true positive as one within/above the threshold for all of the five metrics. We use this to define average precision $\mathrm{AP}(\delta_b, \delta_V, \delta_r, \delta_t, \delta_c)$. To better understand performance limitations, we consider variants where we relax one of these predicates (indicated by a $\cdot$).

4.3. Analyzing 3D Object Prediction

This is the first work that attempts to predict this representation from images and so many design decisions along the way were not obvious. We therefore study the 3D prediction model in isolation to analyze the impact of these approaches. This avoids mixing detection and shape prediction errors, which helps remove confounding factors; it is also the setting in which all other voxel prediction approaches have been evaluated historically.

Qualitative Results. We first show some predictions of the method using ground-truth boxes in Figure 3. Our approach is able to obtain a good interpretation of the image in terms of a scene and set of objects.

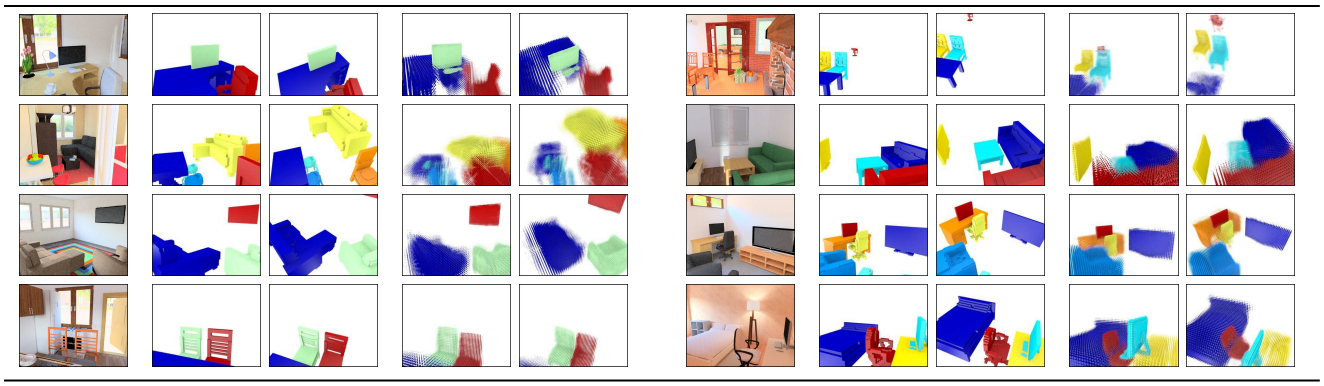

Figure 3: Predicted 3D representation using ground-truth boxes. Left: Input RGB image. Middle (2^{nd} and 3^{rd} column): Two views of ground-truth 3D configuration of the objects in the scene. The first view corresponds to the camera view and the second to a slight rotation towards the top. Right (4^{th} and 5^{th} column): The same two views of our predicted 3D structure. We visualize the predicted object shape by representing each voxel as a cube with size proportional to its occupancy probability and then transform it according to the predicted scaling, rotation and translation. The colors associate the corresponding ground-truth and predicted objects.

Table 1: Performance of predictions on SUNCG with ground-truth boxes: We report the performance of the base network (quaternion classification, use of context) and its variants. We measure the median performance across the 3D code parameters and also report the fraction of data with performance above/error below certain thresholds. See text for details on evaluation metrics.

	Shape		Rotation		Translation		Scale	
method	%(IoU $>$ 0.25)	Med-IoU	%(Err $<$ 30)	Med-Err	%(Err $<$ 1m)	Med-Err	%(Err $<$ 0.5)	Med-Err
Base	59.5%	0.31	75.2%	5.44	90.7%	0.38	85.5%	0.15
Base - context	54.4%	0.27	69.3%	7.69	85.4%	0.47	82.6%	0.19
Regression	58.4%	0.31	48.1%	31.87	88.4%	0.38	86.1%	0.14
Base + decoder finetuning	70.7%	0.41	74.6%	5.28	87.3%	0.42	85.1%	0.15

Comparisons. We report comparisons to test the importance of various components. We begin with a base model (*Base*) from which we add and remove components. This base model is trained using the losses and features described previously, but the decoder set to the autoencoder's decoder.

We first experiment with shape prediction. We add decoder fine-tuning to get (*Base + Decoder Finetuning*), which tests the effect of fine-tuning the decoder. We also try a retrieval setup (*Retrieval*); rather than use the decoder, we retrieve the nearest shape in the shape embedding space.

We then evaluate our features and losses. We try (*No Context*) in which we use *only* the ROI-pooled features; this tests whether context, in the form of bounding box coordinates and full image features, is necessary. Since our classification approach to rotation prediction may seem non-standard, we try an antipodal regression loss for estimating $\mathbf{q}$. We normalize the prediction $\hat{\mathbf{q}}$ and minimize $\min(||\mathbf{q} - \hat{\mathbf{q}}||, ||\mathbf{q} + \hat{\mathbf{q}}||)$.

Quantitative Results. We plot cumulative errors over fractions of the data in Figure 4 and report some summary statistics in Table 1. For the task of shape inference, we observe that fine-tuning the decoder improves performance. The alternate method of retrieval using a shape embedding yields some accurate retrievals, but is less robust to uncertainties, and incurs large errors for many instances. We note that as SUNCG dataset has a common set of 3D objects across all scenes, this retrieval performance can be further improved by explicitly learning to predict a model index. However, this approach would still suffer from large errors in case of incorrect retrieval, and not be generally applicable.

We observe that classification outperforms regression for predicting rotation. We hypothesize that this is because classification handles multi-modality (*e.g.* whether a chair is front- or back-facing) better. Additionally, classification has systematically different failure modes than regression: as compared to a nearly uniform set of errors from regression, the model trained with classification tends to be either very accurate or off by 90° or 180° degrees, corresponding to natural ambiguities. Finally, having context features is consistently important for each error metric, in particular for inferring absolute translation and scale, which are hard to infer from a cropped bounding box.

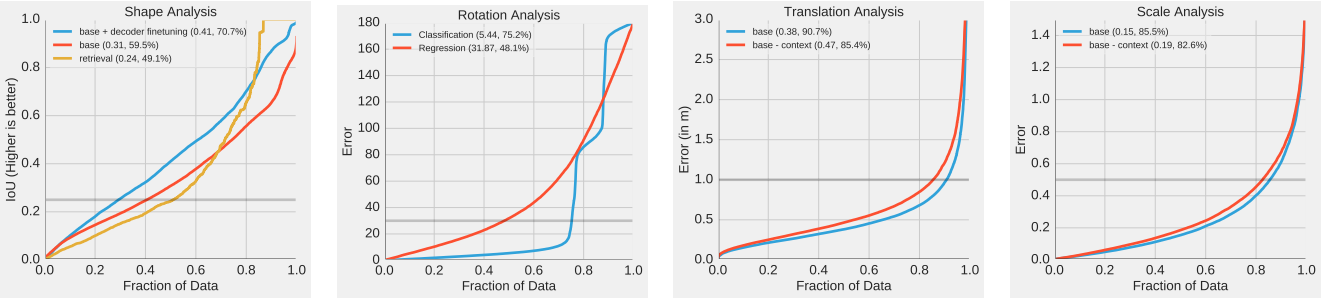

Figure 4: Analysis of the prediction performance across shape, rotation, translation and scale prediction. To compare alternative approaches, we plot the error (or IoU for shape) against the fraction of data up to the threshold. The plot legends also report the median value as well as fraction of data with error lower (or IoU higher) than the threshold depicted by the gray line.

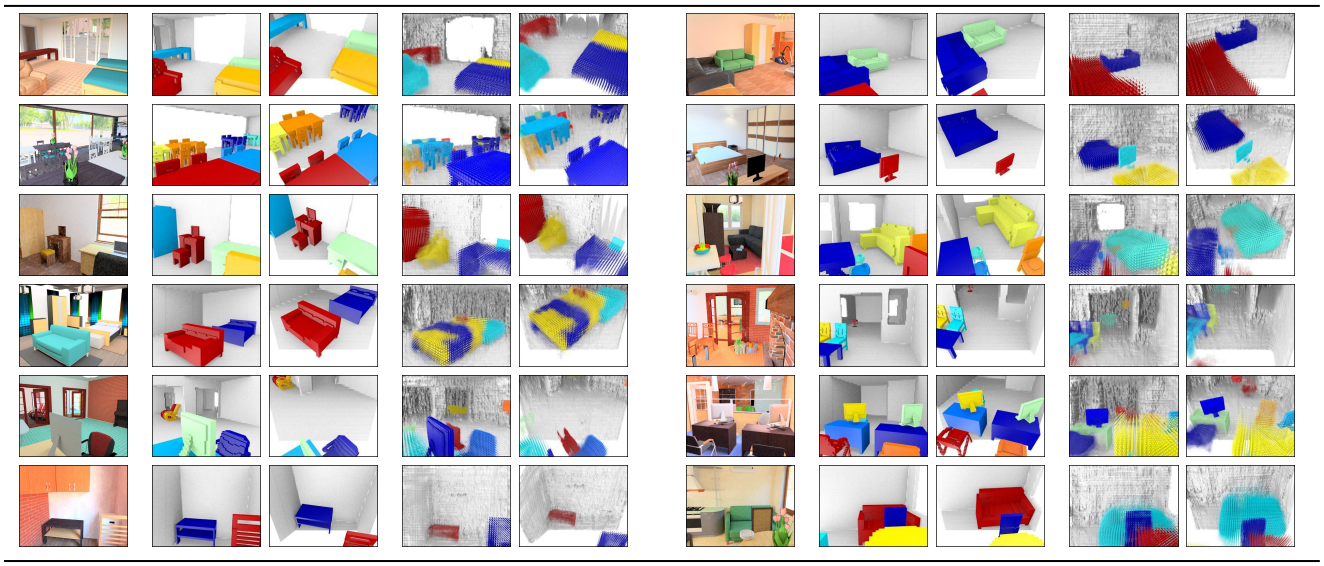

Figure 5: Predicted 3D representation from an unannotated RGB image. Left: Input image. Middle (2^{nd} and 3^{rd} column): Two views of ground-truth 3D configuration of the objects in the scene. Right (4^{th} and 5^{th} column): The corresponding two views of our predicted 3D structure. The colors only indicate a grouping of the predicted points and the coloring is uncorrelated between the prediction and the ground-truth. We observe that we can infer the 3D representations despite clutter, occlusions *etc*.

4.4. Placing Objects in Scenes

Having analyzed the factors of performance for 3D object prediction with known 2D bounding boxes, we now analyze performance on the full problem including detection.

We report in Figure 7 some variants of average precisions on the SUNCG test set for our approach. We obtain an average precision of 40.3% for the full 3D prediction task $AP(0.5, \frac{\pi}{6}, 1, 0.5, 0.25)$. This is particularly promising on our challenging task of making full 3D predictions in cluttered images of scenes from a single RGB image. We also report variants of the AP when relaxing one constraint at a time Figure 7 (left).

Figure 5 visualizes the output of our detector on some validation images from the SUNCG dataset. We show the input RGB image, ground truth and predicted objects from the current view and an additional view (obtained by rotating the camera up about a point in the scene). We observe some interesting error modes, *e.g.* duplicate detections in 3D space despite the underlying boxes not being classified as duplicates via 2D non-max suppression.

4.5. Comparing Scene Representations

We have proposed a new way of representing the 3D structure of scenes, and so one might ask how it compares to the alternatives in use, per-pixel depth or a single voxel grid. As has been argued throughout the paper, our representation is qualitatively different and captures aspects that are missing in the others: as shown in Figure 6), voxel grids and depthmaps present an undifferentiated array of surfaces and

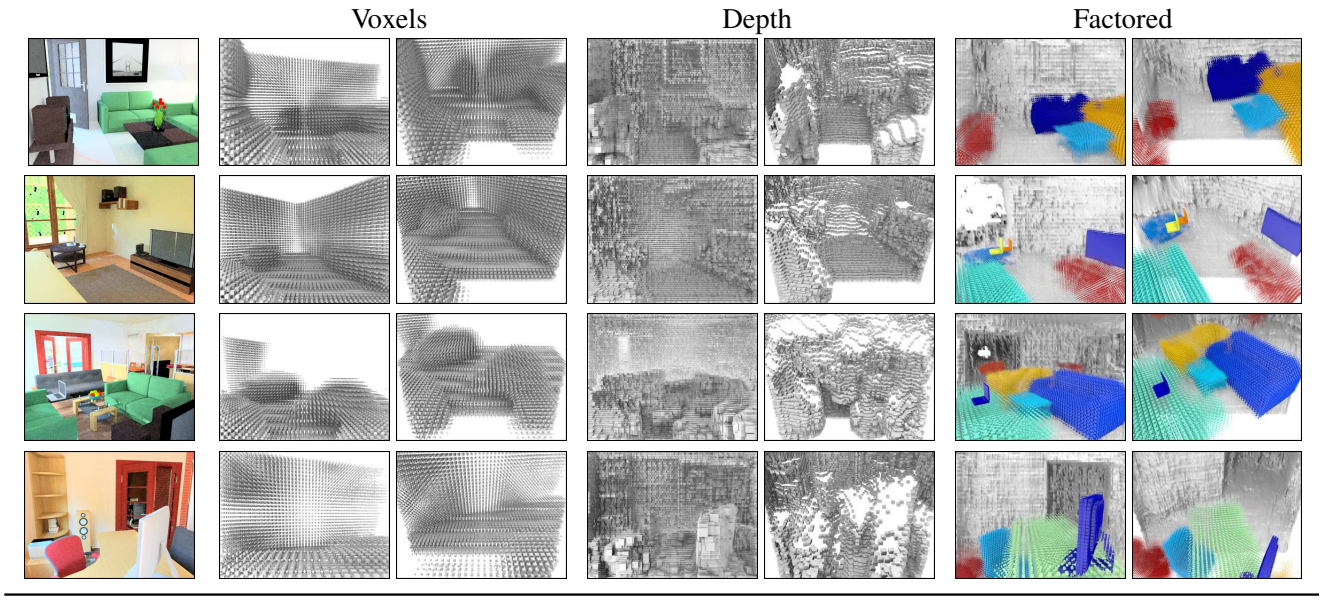

Figure 6: A visualization of the proposed (*Factored*) representation in comparison to (*Voxels*) a single voxel grid and (*Depth*) a depthmap. For each input image shown on the left, we show the various inferred representations from two views each: a) camera view (left), and b) a novel view (right).

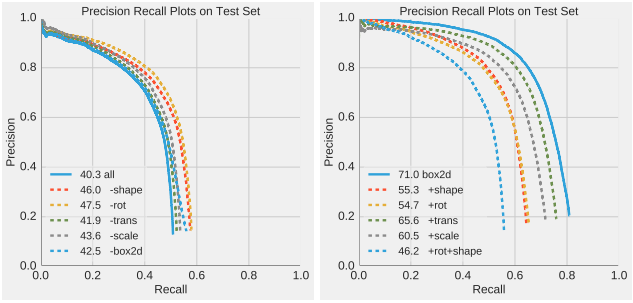

Figure 7: Detection Performance on SUNCG Test Set: We plot PR curves for our method on the SUNCG test set under different settings. Left: PR curve for full 3D prediction (denoted by 'all') and its variants when relaxing one condition at a time. Right: 2D bounding box PR curve (denoted by 'box2d') and its variants when adding one additional constraint at a time. The average precision for each setting is indicated in the legend.

volumes whereas ours represents a world of objects. However, we also quantitatively evaluate this. Each representation (depth, voxels, factored) is trained on different tasks. We study how well each representation solves the tasks being solved by the other representations. While each representation will perform the best at the specific task that it was trained for, a versatile representation will also work reasonably well on the others' tasks. Our experiments below show that our factored representation is empirically more versatile than these two other popular 3D representations.

Other representations.

Per-pixel depth representations estimate the depth (or equivalently disparity *i.e.* inverse depth) for each pixel in the image. We train this representation the same way we train our layout prediction module as described in Section 3.1 on the SUNCG dataset, except instead of predicting the amodal disparity for scene surfaces we make predictions for all pixels in the image as would be observed from a depth sensor.

Full scene voxel representations use occupancy of voxels to represent the scene. We use $64 \times 32 \times 64$ voxels each of size $8cm \times 8cm \times 8cm$ to represent the scene. These voxels are expressed in the camera coordinate frame.

Quantitative results. We show quantitative results in Figure 8. We plot the cumulative distribution for various performance metrics (described below).

Explaining Visible Depth: We obtain a point cloud for the scene from each of these representations (for depth, we backproject points in space using the camera matrix, for voxels we use the point at the center of the voxel). We measure the average distance of points in the predicted point cloud to points in the ground truth point cloud obtained by back projecting the ground truth depth image.

Explaining Scene Voxels: We obtain voxel occupancy from each of the representations (depth is converted to voxel occupancy by checking if any back-projected point lies within a voxel). We measure intersection over union for the output voxel occupancy with the ground truth voxel occupancy.

Our factored representation involves two tasks: reasoning about the objects and the scene surfaces.

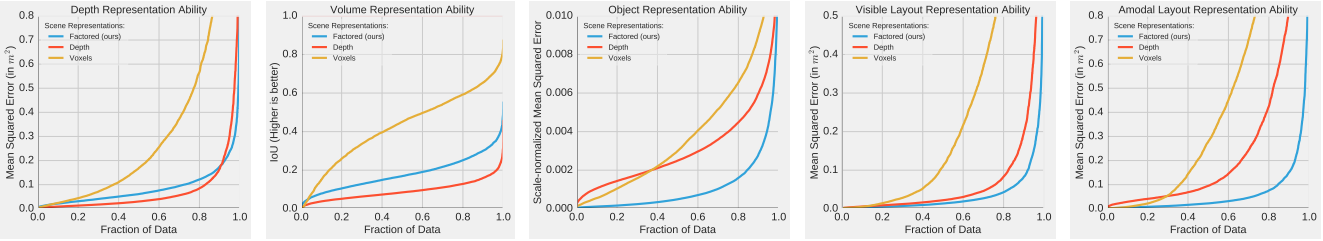

Figure 8: Analysis of the ability of various representations to capture different aspects of the whole scene. We compare our proposed factored representation against voxel or depth-based alternatives and evaluate their ability to capture the following aspects of the 3D scene (from left to right): a) Visible depth, b) Volumetric occupancy, c) Individual objects, d) Visible depth for scene surfaces (floor, walls *etc*.), and e) Amodal depth for scene surfaces. See text for a detailed discussion.

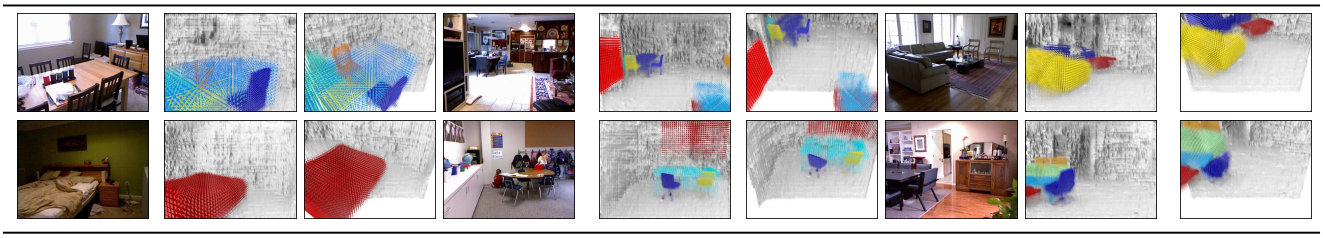

Figure 9: Results on NYU dataset. We show the results of our model trained using synthetically rendered data on real, unannotated images from the NYU dataset. Left: Input RGB image. Middle and Right: Two views of predicted 3D representation.

Explaining Objects: To measure this we align the ground truth object point clouds (obtained by sampling points at center of occupied voxels) to point clouds obtained from the three representations (in the same manner as above). We use iterative closest point to align and report the final fitness value (mean squared distance normalized with respect to the object size). We compute this at instance level for the six categories that we study.

Explaining Layout: We measure how well depth corresponding to the scene layout surfaces is explained and consider two cases: modal scene surfaces (Figure 8(d)) and amodal scene surfaces (Figure 8(e). This metric is similar to the visible depth evaluation described above except it appropriately adjusts the ground truth to focus only on layout (i.e., walls/floors/ceiling).

We observe that indeed each representation excels at the task they were specifically trained for. However our representation consistently shows much better generalization to the other tasks compared to other the other representations. Additionally, even though the visible layout can, in principle, be equally well described using a depth image, our representation works better at predicting visible layout as compared to the full image depth prediction baseline, showing the merit of factored modeling of scene composition.

4.6. Results on NYU

We also tested our models trained on the SUNCG dataset on images from the NYU dataset (Note we only use the RGB image to obtain these results). Figure 9 visualizes the output of our models on NYU Test set images. We obtain these visualizations by running our model with 2D bounding box proposals from [2]. Despite being trained on synthetic data, we are able to obtain a good interpretation of the scene.

5. Discussion

We argue that the representation that one should infer to understand the structure of a 3D scene should be factored in terms of a small number of components: a scene layout, and individual objects, each in turn explained in terms of its shape and pose. We presented a learning based system capable of inferring such a 3D representation from a single image. However, this is only a small step towards to goal of inferring 3D scene representations and that many challenges remain. In particular, we do not reason about the physics and support relationships of the predicted scenes. Additionally, we rely on synthetically rendered data with associated ground-truth for training which limits the performance on real data. However, we hope that parallel efforts in the vision community on more realistic renderings [22], leveraging weaker supervision [36], or scaling up real datasets [6] will help bridge this gap.

Acknowledgements. This work was supported in part by Intel/NSF VEC award IIS-1539099, NSF Award IIS-1212798, and the Google Fellowship to SG. We gratefully acknowledge NVIDIA corporation for the donation of Tesla GPUs used for this research.

References

[1] https://shubhtuls.github.io/factored3d/supp.pdf. 2, 4

[2] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014. 8

[3] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic. Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *CVPR*, 2014. 2

[4] A. Bansal, B. Russell, and A. Gupta. Marr revisited: 2D-3D alignment via surface normal prediction. In *CVPR*, 2016. 1, 2

[5] 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 *ECCV*, 2016. 1, 2, 4

[6] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, 2017. 8

[7] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014. 1, 2

[8] S. Fidler, S. Dickinson, and R. Urtasun. 3D object detection and viewpoint estimation with a deformable 3D cuboid model. In *NIPS*, 2012. 2

[9] D. F. Fouhey, A. Gupta, and M. Hebert. Data-driven 3D primitives for single image understanding. In *ICCV*, 2013. 1, 2

[10] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016. 1, 2, 3

[11] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 3

[12] A. Gupta, A. A. Efros, and M. Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV*, 2010. 2

[13] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik. Aligning 3D models to RGB-D images of cluttered scenes. In *CVPR*, 2015. 2

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4

[15] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009. 2

[16] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *CVPR*, 2005. 2

[17] H. Izadinia, Q. Shan, and S. M. Seitz. IM2CAD. In *CVPR*, 2017. 2

[18] D. C. Lee, A. Gupta, M. Hebert, and T. Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *NIPS*, 2010. 2

[19] Y. Li, H. Su, C. R. Qi, N. Fish, D. Cohen-Or, and L. J. Guibas. Joint embeddings of shapes and images via cnn image purification. *TOG*, 2015. 2

[20] J. J. Lim, H. Pirsiavash, and A. Torralba. Parsing ikea objects: Fine pose estimation. In *ICCV*, 2013. 2

[21] 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 *CVPR*, 2016. 2

[22] J. McCormac, A. Handa, S. Leutenegger, and A. J.Davison. Scenenet RGB-D: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation? In *ICCV*, 2017. 8

[23] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis. 6-dof object pose from semantic keypoints. In *ICRA*, 2017. 2

[24] L. G. Roberts. *Machine Perception of Three-Dimensional Solids*. PhD thesis, MIT, 1963. 2

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 4

[26] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3D scene structure from a single still image. *TPAMI*, 2009. 1, 2

[27] A. G. Schwing, S. Fidler, M. Pollefeys, and R. Urtasun. Box In the Box: Joint 3D Layout and Object Reasoning from Single Images. In *ICCV*, 2013. 2

[28] A. Shrivastava and A. Gupta. Building part-based object detectors via 3D geometry. In *ICCV*, 2013. 2

[29] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012. 4

[30] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, 2017. 1, 2, 4

[31] 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 *ICCV*, 2015. 3

[32] S. Tulsiani and J. Malik. Viewpoints and keypoints. In *CVPR*, 2015. 2, 3, 4

[33] 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 *NIPS*, 2016. 1, 2

[34] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond pascal: A benchmark for 3D object detection in the wild. In *WACV*, 2014. 2

[35] 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. *CVPR*, 2017. 4

[36] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017. 8

[37] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 4