Vision, Modeling, and Visualization (2016)
M. Hullin, M. Stamminger, and T. Weinkauf (Eds.)

Scene Structure Inference through Scene Map Estimation

Moos Hueting[1] Viorica Pătrăucean[2] Maks Ovsjanikov[3] Niloy J. Mitra[1]

[1]University College London [2]University of Cambridge [3]LIX, École Polytechnique

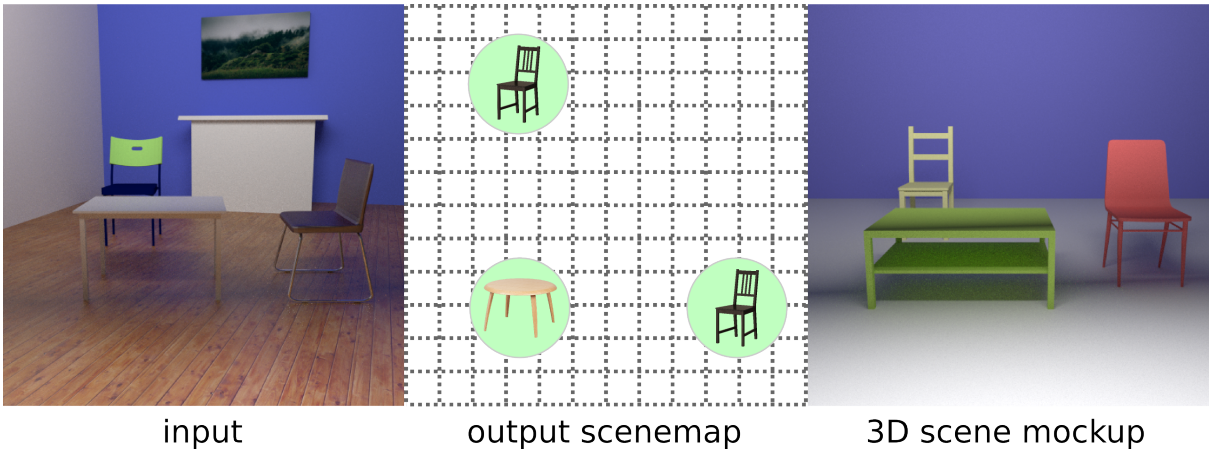

Figure 1: *Given a single RGB image (left), we propose a pipeline to generate a scene map (middle) in the form of a floorplan with grid locations for the discovered objects. The resulting scene map can then be potentially used to generate a 3D scene mockup (right). Please note that our system does not yet support pose estimation; hence in the mockup the objects are in default front-facing orientations.*

Abstract

Understanding indoor scene structure from a single RGB image is useful for a wide variety of applications ranging from the editing of scenes to the mining of statistics about space utilization. Most efforts in scene understanding focus on extraction of either dense information such as pixel-level depth or semantic labels, or very sparse information such as bounding boxes obtained through object detection. In this paper we propose the concept of a scene map*, a coarse scene representation, which describes the locations of the objects present in the scene from a top-down view (i.e., as they are positioned on the floor), as well as a pipeline to extract such a map from a single RGB image. To this end, we use a synthetic rendering pipeline, which supplies an adapted CNN with virtually unlimited training data. We quantitatively evaluate our results, showing that we clearly outperform a dense baseline approach, and argue that scene maps provide a useful representation for abstract indoor scene understanding.*

1. Introduction

Our ability of reasoning from visual input is vital, both to the constant and continuous analysis of our surroundings, as well as to the formation of a correct response to this analysis. If we are to create intelligent systems capable of navigating the intricacies of the real world as well as human beings, we need to find ways of recreating this impressive mental ability. Hence, not surprisingly, indoor scene understanding has received significant research attention in both computer graphics and vision.

A core subtask of indoor scene understanding is *scene structure inference*, i.e., deducing the presence and the locations of individual objects composing the scene. Given a single image, as humans, we can in most cases tell the class of the objects and their relative positions in the scene. Of course, from this information a lot can be inferred, such as an unobstructed path through the room, scale, and scene type (a room containing a bed and a chair is likely to be a bedroom, while a room containing a desk and an executive chair is likely to be an office). Such floorplan-level information thus provides a compact and useful summary about the nature and the structure of the scene (see Figure 1).

DOI: 10.2312/vmv.20161341

One possible solution to inferring such a scene floorplan from a single image is to merge state-of-the-art solutions to both semantic segmentation and depth estimation to define the location of all objects in the scene. Such an approach has several disadvantages. Firstly, it is not trivial to delineate the boundaries of the individual objects in the image, as most of the existing semantic segmentation approaches do not produce instance-aware segmentation [SSF14]. Secondly, both semantic segmentation and depth estimation model each pixel individually. Fundamentally, as we are only interested in the relative location of the objects, the intermediate steps that involve pixel-level labeling can introduce a significant source of error affecting holistic scene understanding.

To circumvent these problems, we propose a novel representation, called a *scene map*. The scene map models the structure of an indoor scene using a collection of grids that mark the location of objects of different classes in a top-down view of the scene (see Figure 2). Importantly, as we target directly the global structure of the scene compared to e.g., extracting placement in world coordinates, low resolution grids suffice for this task. This limits the number of variables necessary to train and estimate in practice to the bare minimum.

In this paper, we present a pipeline for estimating the scene map from a single image. At its heart lies a convolutional neural network, based on the successful VGG architecture [SZ14]. As training data for scene maps is scarce, we create a rendering pipeline that synthesizes scenes and renders them on the fly, supplying the network with a virtually unlimited amount of training data. Using this synthetic training data, the network is trained end-to-end. The pipeline's performance is compelling, with 52% of models being located within one grid cell of their ground truth location.

We compare our method with a baseline that combines state-of-the-art semantic segmentation and single frame depth estimation. Our evaluation shows that the scene map representation gives more accurate results for this task, while needing to solve for a significantly fewer variables than its baseline counterparts. We conclude with discussion of limitations and future work.

Our main contributions thus include:

- Introducing the scene map as a representation for holistic scene understanding;
- suggesting a method for synthesizing scenes together with their scene maps by exploiting a 3D model collection, and using this data to train a convolutional neural network; and
- proposing a method for inferring the scene map given a single frame as input, using our learning pipeline.

2. Related Work

Image-based scene understanding is one of the core areas of both computer graphics and computer vision with a wide variety of approaches that have been proposed over the years. Below we only review the works most closely related to ours, which in particular try to combine object detection with 3D localization for indoor scene labeling, and using synthetic data for training methods in scene understanding.

Semantic segmentation. Many traditional approaches for scene understanding are based on semantic segmentation, which tries to associate class labels to pixels in the image (see [GH14] for an overview of related methods). Most recently, successful techniques heavily exploit training data to guide semantic segmentation (e.g., [CCMV07, CPK*14, NHH15] among many others). Moreover, some recent approaches have used synthetic (rendered) data to augment the training set resulting in more accurate labeling [HPB*15]. Unlike these methods, however, our goal is not to associate class labels to image pixels, but to directly output a scene map, which summarizes the objects in the image *in scene coordinates*. In this way, our approach is related to techniques that estimate depth together with semantics [EF15], although we avoid the error-prone depth estimation step by training on the scene maps directly.

Scene mockups. A number of methods have also been proposed for high-level scene understanding and labeling, by exploiting additional depth information available from RGB-D sensors [SX14, GAGM15, SLX15]. Although our goals are similar, we only use 2D image information at test time, and exploit rendered synthetic scenes for training. Perhaps most closely related to our method is a very recent technique by Bansal et al. [BRG16] who use a database of 3D models and retrieve the closest model to a *given* bounding box in the image. In addition, they do dense normal estimation first, which again introduces additional complexity and a potential source of inaccuracies.

Joint 2D-3D analysis. Our work is also closely related to the recent methods that exploit the large databases of 3D models, such as Trimble 3D Warehouse, to facilitate image analysis. Most notably, 3D model collections have been used for joint retrieval [HOM15], single-view reconstruction [HWK15], object detection [AME*14, MRA15], view-point estimation [SQLG15], scene parsing [ZZ13], or even for learning generative models for object synthesis [GFRG16]. Model collections are particularly useful as a source of additional training data that can be incorporated into learning algorithms for labeling [HPB*15] or pose estimation tasks [CWL*16] among many others [WHC*15, WSK*15, BMM*15]. In this area, our work is most closely related to methods that use 3D data for scene understanding [LZW*15, KIX16].

Unlike these methods, however, our main goal is to use synthetic data to train a method for semantic scene understanding that directly produces a summary of the presence and positions of the objects *in scene coordinates*. This allows us to avoid the unnecessary intermediate computations, such as depth or normal-estimation, or floor-plane detection, while resulting in highly informative and concise scene summaries.

3. Method

Our method infers scene structure from a single RGB image. It does so by learning a mapping from the input to a new representation called a *scene map* through the use of a deep neural network. We will first discuss this new representation, and then detail the network architecture at the core of our method. Finally, we explain our synthetic rendering pipeline, which feeds the network with an unlimited supply of training data, which helps to offset the lack of real training data for this purpose.

Figure 2: *A scene map describes the scene on a per-class basis from a top-down view corresponding to an input RGB image. A white square indicates the presence of an instance of that particular class at that location. Here we show the groundtruth scene map, while our result can be seen in Figure 7, bottom row.*

3.1. Scene Map

Our system takes as input a single RGB image, and outputs a top-down view of the scene called a *scene map*. Intuitively, the scene map provides a two-dimensional summary of the objects present in the scene and their relative positions in a way that is similar to a floor-plan. The two coordinates of the scene map correspond to the x and y coordinates of the plane parallel to the floor of the given indoor scene, and the values stored at a particular coordinate correspond to the objects present at that position. Importantly, the scene map completely removes the third coordinate (height), and only represents the floor implicitly by using it as a frame of reference for other objects. As we demonstrate below, such a reduced representation greatly facilitates the inference and learning tasks, while still providing a very useful summary of the overall scene structure.

More precisely, assuming that the scenes contain objects belonging to N different classes, a scene map S consists of grids G_i of resolution $r \times r$, with $i \in \{1 \dots N\}$, giving one grid per class. Each grid is represented as a binary matrix, which marks the locations of all instances of any class in the scene; see Figure 2. This representation is inspired by the popular *occupancy grid* representation commonly used in robotics applications for 3D mapping [TBF05], where the 3D environment is modeled as an evenly-spaced field of binary random variables, taking the value 1 when an obstacle is present at the corresponding location. Thus, a scene map can be considered as a spatial-semantic occupancy grid, with 2 dimensions reserved to spatial coordinates and the third to class identity.

The scene map is of limited resolution by design. As we are interested in the general layout of a scene, a margin of e.g., 30cm in the placement of an object could be acceptable. By assuming such a margin, the number of variables in the placement problem ($r \times r \times N$) is significantly reduced compared to using a fine grid, or to modeling the problem in the original pixel space ($w \times h \times N$). This type of simplification is encountered equally in computer vision applications that convert regression into classification: e.g., in [ACM15] where the aim is to predict ego-motion encoded as a rotation-translation movement. Instead of regressing to precise (continuous) angle and translation values, the problem is converted into a classification task by binning each movement into a fixed (discrete) number of ranges of movement magnitude. This choice results in a sensible trade-off between accuracy and complexity in problems where very precise predictions are not mandatory.

In our setup, the scene map is designed to encode a square area on the floor of the scene in front of the camera of 6m $\times$ 6m in size. This is large enough to accommodate more than 95% of the scenes in the SUNRGB-D dataset, and can easily fit the average UK room size [Wil10]. We use grids of size 16×16, resulting in a grid cell size of 37.5cm.

3.2. Scene Map Inference Overview

Our main goal is to compute the scene map representation from a single input RGB image. For this we follow a data-driven approach that has been shown to be effective for a wide variety of image processing tasks. Namely, we train a Convolutional Neural Network (CNN) that, given a single image, tries to output its scene map representation directly, without estimating any low-level attributes such as depth or pixel-wise class labels. One challenge with adopting this approach, however, is that it requires a large amount of training data to be successful, due in part to the large number of variables that typically need to be estimated. Unfortunately, there is no existing sufficiently large dataset that contains ground truth scene map labelings (e.g., the recent SUNRGB-D dataset [SLX15] contains approximately 10000 images). To overcome this issue, we train our network with scenes that we synthesize on the fly by exploiting an existing 3D model collection [LPT13] and varying the composition of the scene and the appearance of the objects using a large texture dataset [PGM*95] using a probabilistic model. In particular, we create a scene synthesis pipeline that uses a rendering approach and a randomized object placement and appearance variation model. This pipeline effectively provides our learning framework with an unlimited source of data that we use to train an adapted CNN for scene map inference. To summarize, our general approach consists of the following key steps:

- Adapting a well-developed CNN architecture for inference of scene maps from single images.
- Constructing a randomized scene synthesis pipeline based on a scene composition model coupled with appearance variation and an efficient rendering method.
- Using our scene synthesis method to train the network by generating a large number of ground truth pairs consisting of an image and its associated scene map.
- Using the trained network to estimate the scene map on a new test image.

Below we describe each of the individual steps of our pipeline and provide the corresponding implementation details.

3.3. Network

To learn the mapping from the RGB image space to the scene map representation, we use a deep neural network that builds upon VGG11 [SZ14], with a few modifications (see Figure 3). Notably, we added batch normalization after each convolutional and fully-connected layer, resulting in a significant decrease in training time [IS15]. The original VGG11 maps the input image to a discriminative feature representation in $\mathbb{R}^{1024}$, then uses a classifier to

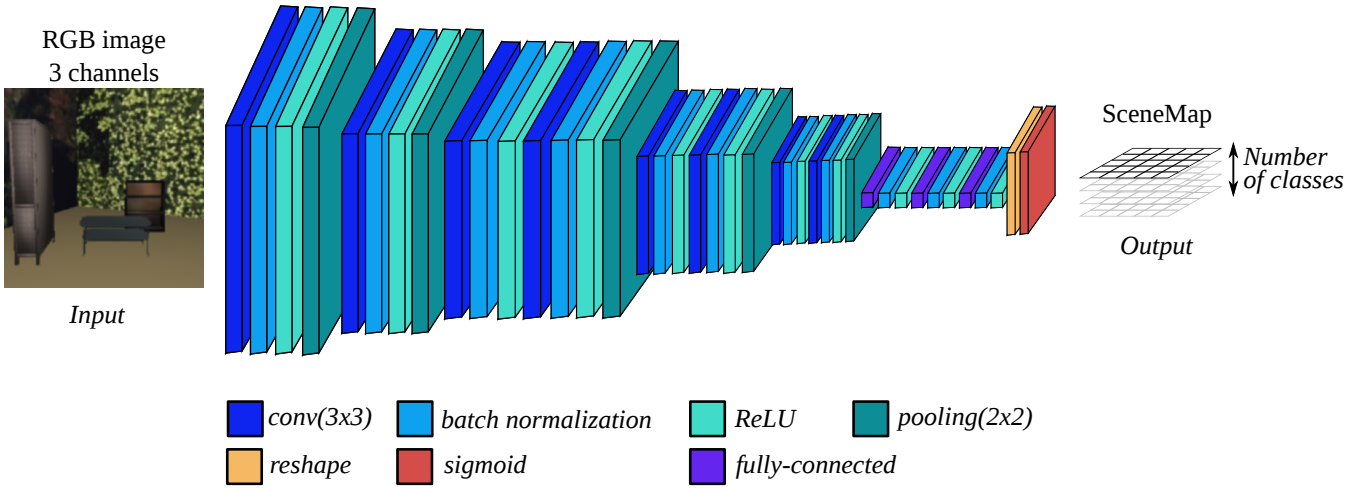

Figure 3: *Our network architecture, based on VGG11.*

predict a class label for each image. Since our problem requires a spatial representation and not a single class label, we remove the classifier and instead reshape this representation to the desired scene map representation of size $r \times r \times N$. Note that most architectures designed for spatial tasks (e.g. for semantic segmentation) use mirrored encoder-decoder networks, enforcing direct correspondences between the feature maps learnt by the encoder and the decoder at each level [NHH15]. But in our case such architectures are not justified, since the input domain (image pixels) is different in resolution and viewpoint from the output domain (grid cells). However, investigating for more adapted architectures for our task constitutes a direction of future work. The result is passed through a sigmoid layer, so that each cell in the grid reflects the likelihood of an object of a given class being present. The overall architecture has 20 million parameters.

Training. We have implemented the proposed network using Torch. The RMSprop optimiser [TH12] was used with an initial learning rate of 10^{-3}, and a learning rate decay of 0.8 after every 10000 iterations. Note that the training is done from scratch, since no pre-trained models are available for VGG11. The training was performed on a multi-GPU system (4 GPUs, 12G memory each), with batch size of 32, and took approximately 10 hours to converge.

3.4. Non-maximum suppression

Each cell within a class grid shows the confidence of the network about the presence of an object of that class at the corresponding spatial location. However, the network is often uncertain about the precise location of an object. This uncertainty is expressed by a spreading of the probability across multiple cells in the vicinity of the actual location. Note that this behavior is justified considering that depth estimation from a single image suffers from scale ambiguity, especially when a certain object has not been seen before. Deep learning approaches for depth estimation from single RGB images [EF15] try to bypass this issue by implicitly learning absolute scale ranges for each object from the large number of training examples. The intra-class scale variability will dictate the range width, and eventually the accuracy that can be obtained; wide range resulting in more uncertainty in the output. A very simple idea to reduce uncertainty and binarize the probability maps would be to use a fixed cut-off value, e.g. 0.5, and deem every cell with an output probability of 0.5 or higher to contain an object of that class. However, we found that performing a max pooling post-processing step, with a 3×3 window, results in sparser, more accurate scene maps than direct thresholding. Hence, we use this approach in our experiments (see Figure 4).

3.5. Rendering pipeline

Training a deep neural network requires large amounts of training data. The largest available dataset for our purpose, SUNRGB-D [SLX15], contains approximately 10000 images with 60000 bounding boxes of 1000 different classes. We have found this not to be

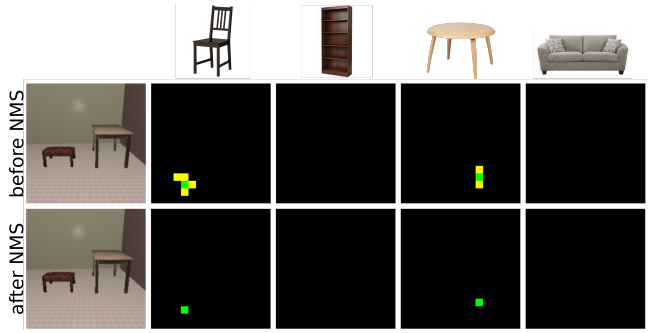

Figure 4: *Result of non-maximum suppression. Yellow cells represent false positives, green cells true positives. The top row shows the scene map after simple thresholding at 0.5. This results in spurious activations around the true location of each object. After non-maximum suppression, these are removed, with only the local maximum (the true positive) being left.*

enough for training a network that generalizes well. To boost our training data numbers, we set up a synthetic rendering pipeline, which renders training pairs of images with the associated scene maps on the fly. This provides the system with an unlimited stream of essentially unique training data (although theoretically two scenes could be identical, the probability of this is vanishingly low). This is an instance of *online learning*, which has self-regularizing capabilities, limiting the risk of overfitting [Bot98].

Data. The rendering pipeline takes a set of class-labeled objects O and textures T as input. In our experiments, we take O as a subset of the IKEA dataset [LPT13]. This dataset contains objects of four classes *chair, shelf, table* and *sofa*, with 16 objects per class. We manually curated all models to have accurate relative scale and to be centered at world origin, with consistent orientation. The texture set T consists of a subset of 136 textures from the VisTex dataset [PGM*95], which we curated manually to be appropriate textures for furniture. Both O and T are separated into training and test sets, using a 75%/25% split, as illustrated in Figure 5.

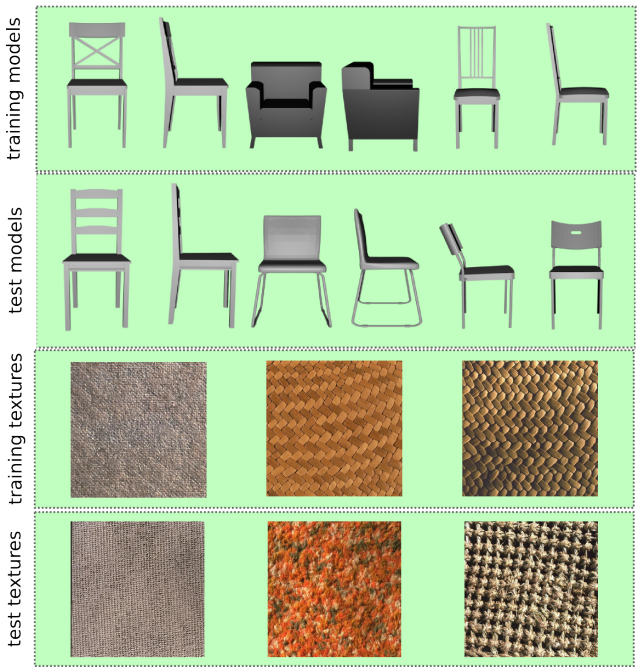

Figure 5: *We split models and textures into training and test sets at a 75%/25% split. This allows us to test how well the network learns the general shape of each class of object.*

Scene generation. When the pipeline is queried for a new training example, a new random scene is generated (see Figure 6). First, a subset o ranging from 2 to 6 objects from O is sampled at random with replacement. These objects are then randomly placed around the world center in a 6×6 meter square. The objects are randomly rotated around the up-vector in increments of 90 degrees. We make sure the objects placed do not collide with each other. Inspection of 100 publicly available indoor photographs shows that one wall is nearly always visible, with a second perpendicular wall being visible approximately 75% of the time. As such, two perpendicular walls

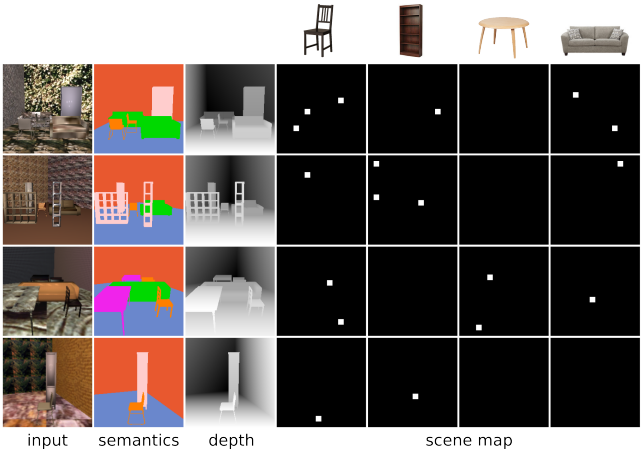

Figure 6: *An illustration of 4 samples from the data generation pipeline. We render RGB, semantic segmentation, depth, and scene map, resulting in an unlimited stream of fully defined training data.*

are placed around the scene, with random, but coherent orientation. Finally, the objects and walls are individually textured by randomly sampling and scaling from our texture set T. Note that by including small texture scales, we mimic textures with repeating patterns.

The camera is placed at a height drawn randomly in the range between $1m$ and $1.8m$ to mimic the range of heights from which most handheld photographs are taken.

Rendering. The generated scenes are then rendered using an OpenGL setup with a simple Phong shading model. For each scene, we also generate the scene map, semantic labeling, and depth ground truth. The latter two are used only for baseline comparison (see Section 4.1). See Figure 6 for some samples from the rendering pipeline.

4. Evaluation

In the preceding sections we proposed the scene map as a representation for holistic scene understanding, reasoning that its low dimensionality removes unnecessary variables from the optimization process compared to dense, pixel-based approaches. Below, we compare our scene map estimation method with a dense approach that combines the output of semantic segmentation with depth estimation, both from single frame RGB input.

4.1. Baseline

Semantic segmentation. The semantic segmentation pipeline we use is a version of [NHH15], using VGG11 [SZ14] as the basis encoder instead of VGG16 (to be comparable with our pipeline in terms of depth), as well as with the fully connected layers removed. The final layer has 6 output maps, one for each class (i.e., chair, shelf, table, sofa) plus two for the wall and floor. We use a spatial cross-entropy loss, classifying each pixel individually. This pipeline is trained from scratch on the same data as our scene map pipeline (see Section 3.5).

Depth estimation. Our depth estimation network is similar, but the final layer outputs just a single map instead of the number of

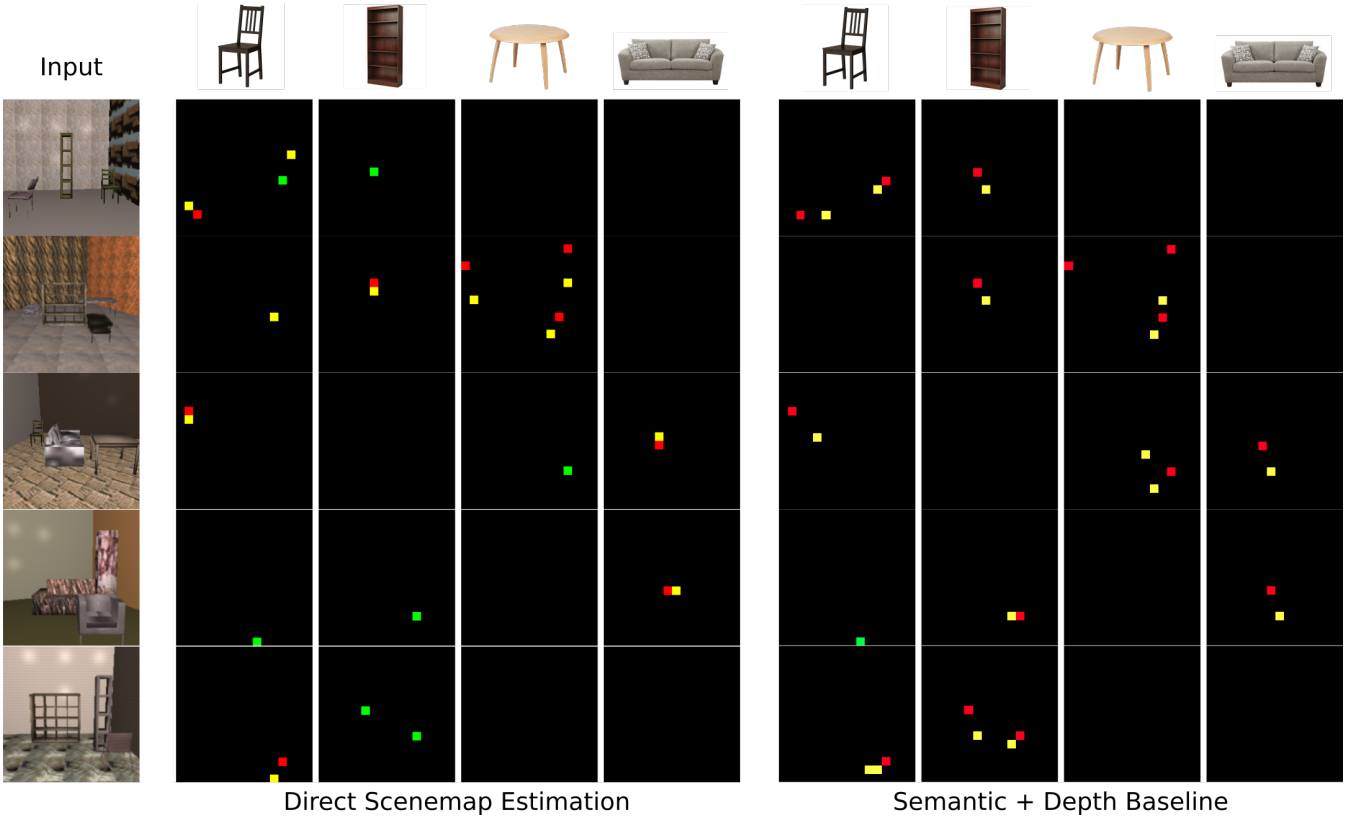

Figure 7: *Left: generated scene map using our pipeline. Each column represents a specific class, each row is one sample. Ground truth is represented using cell color: green grid cells indicate true positives, yellow grid cells indicate false positives, red grid cells indicate false negatives. Right: scene map generated using the semantic segmentation + depth estimation baseline.*

classes as before. We use a log mean squared error loss on the output [EF15].

Combining outputs. We convert the output of the above two networks into a scene map. First, connected components are extracted from the semantic segmentation. Then, for each component, we find the average depth using the estimated depth map. Using the known focal length of the camera we then compute the 3D location of the center of each component, and project this into the scene map.

Performance of networks. By themselves, these networks show high performance in their respective tasks on this data. On the test models and test textures, the semantic segmentation network shows an accuracy of 96.5%, and the depth network an rMSE of 17cm. These unusually high numbers (for reference, [NHH15] report an accuracy of 72.5% in the original paper) are in part due to the unlimited amount of training data our synthetic rendering pipeline provides the network. Moreover, the data used in [NHH15] likely has higher variability and noise than our data, as they come from real photographs instead of synthetically rendered scenes, and hence are more challenging.

4.2. Training vs. test

As discussed in Section 3.5, all training data is generated synthetically, resulting in images very unlikely to be seen twice, but the models used for generating the images are seen many times over. We will evaluate the network both on images generated using these same models, as well as images generated using the models in the test set. Both scenarios are plausible: one can imagine the case where the types of models used in the scene are known in advance, or the case where only the classes are known.

4.3. Comparison

In Figure 7, we show the output of our pipeline, as well as the output of the baseline. Our method shows a clear advantage in performance over the baseline method. Note that although our method does not always find the perfect location, in virtually all cases the presence of an object is detected. The baseline sometimes misses an object entirely, and often activates two cells for a single object.

In Table 1 we compare the accuracy of our method with the baseline quantitatively. Performance is evaluated on both training models as well as test models. Aside from the true and false positive rates, we also report the performance when "one-off" and "two-off" errors are counted as correct (i.e., an object detection one or two cells away

Method	Model set	TPR	TPR+1	TPR+2	FPR
Baseline	Train	0.03	0.20	0.47	0.0037
Ours	Train	**0.24**	**0.66**	**0.82**	**0.0029**
Baseline	Test	0.02	0.19	0.43	0.0043
Ours	Test	**0.15**	**0.52**	**0.71**	**0.0031**

Table 1: *Comparison of our method with the semantic segmentation + depth estimation baseline. TPR is true positive rate, TPR+1 and TPR+2 are true positive rates when respectively off-by-one and off-by-two errors are allowed. FPR is false positive rate. Note that as most of the grid is empty, false positive rates are very low.*

from the ground truth is still counted). Our method significantly outperforms the baseline on all settings. It is interesting to note that for the baseline there is not much difference in performance between the training and test models. For our own method, the decrease in performance from training to test is more significant, while still outclassing the baseline by a compelling margin.

4.4. Effect of object density

To test our pipeline's scalability with respect to the number of objects in the scene, we tested two scenarios where the this number was respectively increased to a range of 6 to 9 objects, and 10 to 15 objects. These scenarios generate scenes with objects very close together, making the task of distinguishing objects more difficult. Sample results can be seen in Figure 8. Clearly, the network is having increasingly more difficulty placing each object at the right location. Moreover, it more often fuses two objects together, resulting in just a single cell being activated. Table 2 shows the quantitative decrease in performance as the number of objects in the scene increases.

Number of objects	TPR	TPR+1	TPR+2	FPR
5-9	0.11	0.42	0.58	0.0044
10-15	0.09	0.32	0.44	0.0053

Table 2: *Effect of object density on performance of our pipeline. Performance decreases with increased scene occupation, but does not dwindle as to be unusable.*

5. Discussion and Conclusion

We have presented a new representation for scene structure called the *scene map*. It reduces the number of parameters necessary for representing scene structure to a minimum, thereby reducing the necessary variables to estimate during optimization. Although the accuracy of the proposed method is limited by design through the size of the grid cells, our output can be directly used for a number of tasks, some of which are detailed below. This is opposed to pixel-wise approaches, which are designed to output accurate predictions, but whose output necessitates non-trivial post-processing to become usable in practice, as shown in our evaluation.

Future work. While we proposed a first pipeline to extract scene maps from single RGB images, several refinements remain to be explored. As it stands, not enough real data is available to train our network from scratch. To extend our method to real images, we plan to use a method for domain adaptation between synthetic and real images (e.g., SUNRGB-D). Moreover, we aim to compare with other

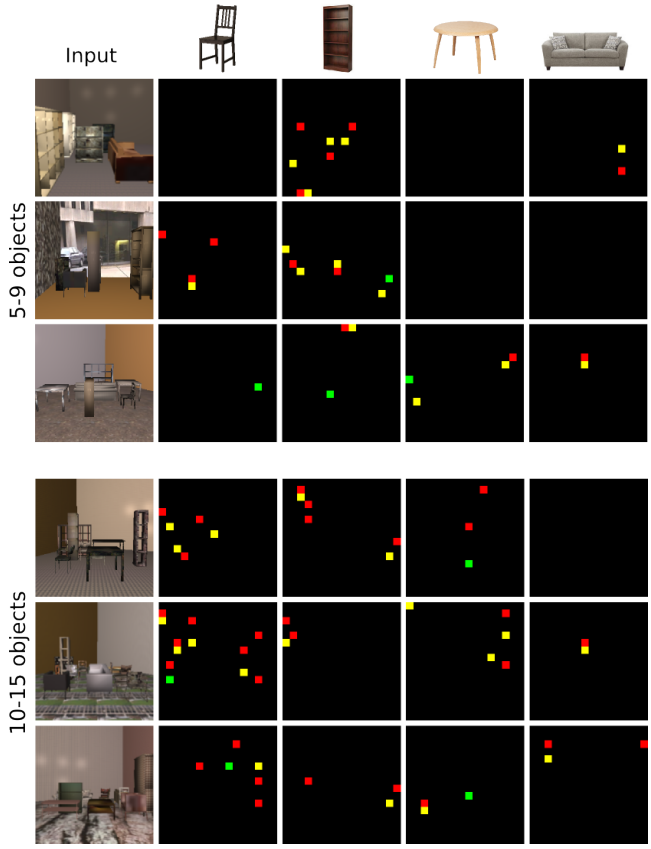

Figure 8: *Results with increasingly dense scenes. Ground truth is represented using cell color: green grid cells indicate true positives, yellow grid cells indicate false positives, red grid cells indicate false negatives. The precise localization of objects becomes more difficult, but in general the presence of objects is still inferred correctly.*

baseline methods. For example, in the current baseline the semantic segmentation step could be replaced by an object detection pipeline (e.g. [RHGS15]). Finally, evaluating on a larger set of classes is needed to show applicability on more varied scenes.

We believe that scene maps extracted from a single image can be directly used for multiple purposes, as they provide a complete summary of the composition and structure of the scene. For example, they open the possibility of automatic retrieval of images with specific scene configurations. In a complementary task, scene maps can help to automatically extract statistics of space utilization from large image datasets [NSF12, SLX15]. Such statistical models could be used for different tasks such as improved scene synthesis and scene type classification. Finally, when combined with in-class model retrieval and a pose estimation pipeline, scene mockups can be potentially generated from scene maps, which in turn can be helpful for architectural visualization and scene relighting.

Acknowledgements This work is in part supported by the Microsoft PhD fellowship program, EPSRC grant number EP/L010917/1, Marie-Curie CIG-334283, a CNRS chaire d'excellence, chaire Jean Marjoulet from École Polytechnique, FUI project TANDEM 2, a Google Focused Research Award, and ERC Starting Grant SmartGeometry (StG-2013-335373).

References

[ACM15] AGRAWAL P., CARREIRA J., MALIK J.: Learning to see by moving. In *ICCV* (2015), pp. 37–45. 3

[AME*14] AUBRY M., MATURANA D., EFROS A. A., RUSSELL B. C., SIVIC J.: Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *CVPR* (2014), pp. 3762–3769. 2

[BMM*15] BOSCAINI D., MASCI J., MELZI S., BRONSTEIN M. M., CASTELLANI U., VANDERGHEYNST P.: Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. In *Computer Graphics Forum* (2015), vol. 34, pp. 13–23. 2

[Bot98] BOTTOU L.: Online algorithms and stochastic approximations. In *Online Learning and Neural Networks*, Saad D., (Ed.). Cambridge University Press, Cambridge, UK, 1998. revised, oct 2012. URL: http://leon.bottou.org/papers/bottou-98x. 5

[BRG16] BANSAL A., RUSSELL B., GUPTA A.: Marr revisited: 2d-3d model alignment via surface normal prediction. In *CVPR* (2016). 2

[CCMV07] CARNEIRO G., CHAN A. B., MORENO P. J., VASCONCELOS N.: Supervised learning of semantic classes for image annotation and retrieval. *IEEE transactions on pattern analysis and machine intelligence 29*, 3 (2007), 394–410. 2

[CPK*14] CHEN L.-C., PAPANDREOU G., KOKKINOS I., MURPHY K., YUILLE A. L.: Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062* (2014). 2

[CWL*16] CHEN W., WANG H., LI Y., SU H., LISCHINSK D., COHEN-OR D., CHEN B., ET AL.: Synthesizing training images for boosting human 3d pose estimation. *arXiv preprint arXiv:1604.02703* (2016). 2

[EF15] EIGEN D., FERGUS R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV* (2015), pp. 2650–2658. 2, 4, 6

[FMR08] FELZENSZWALB P., MCALLESTER D., RAMANAN D.: A discriminatively trained, multiscale, deformable part model. In *CVPR* (2008), pp. 1–8.

[GAGM15] GUPTA S., ARBELÁEZ P., GIRSHICK R., MALIK J.: Aligning 3d models to rgb-d images of cluttered scenes. In *CVPR* (2015), pp. 4731–4740. 2

[GFRG16] GIRDHAR R., FOUHEY D. F., RODRIGUEZ M., GUPTA A.: Learning a predictable and generative vector representation for objects. *arXiv preprint arXiv:1603.08637* (2016). 2

[GH14] GOULD S., HE X.: Scene understanding by labeling pixels. *Communications of the ACM 57*, 11 (2014), 68–77. 2

[Gir15] GIRSHICK R.: Fast r-cnn. In *ICCV* (2015), pp. 1440–1448.

[HOM15] HUETING M., OVSJANIKOV M., MITRA N. J.: Crosslink: joint understanding of image and 3d model collections through shape and camera pose variations. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 233. 2

[HPB*15] HANDA A., PATRAUCEAN V., BADRINARAYANAN V., STENT S., CIPOLLA R.: Scenenet: Understanding real world indoor scenes with synthetic data. *arXiv preprint arXiv:1511.07041* (2015). 2

[HWK15] HUANG Q., WANG H., KOLTUN V.: Single-view reconstruction via joint analysis of image and shape collections. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 87. 2

[IS15] IOFFE S., SZEGEDY C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML* (2015), JMLR Workshop and Conference Proceedings, pp. 448–456. 3

[KIX16] KOHLI Y. Z. M. B. P., IZADI S., XIAO J.: Deepcontext: Context-encoding neural pathways for 3d holistic scene understanding. *arXiv preprint arXiv:1603.04922* (2016). 2

[KSES14] KHOLGADE N., SIMON T., EFROS A., SHEIKH Y.: 3d object manipulation in a single photograph using stock 3d models. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 127.

[LPT13] LIM J. J., PIRSIAVASH H., TORRALBA A.: Parsing IKEA objects: Fine pose estimation. In *ICCV* (2013), pp. 2992–2999. 3, 5

[LSD15] LONG J., SHELHAMER E., DARRELL T.: Fully convolutional networks for semantic segmentation. In *CVPR* (2015), pp. 3431–3440.

[LSQ*15] LI Y., SU H., QI C. R., FISH N., COHEN-OR D., GUIBAS L. J.: Joint embeddings of shapes and images via cnn image purification. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 234.

[LZW*15] LIU Z., ZHANG Y., WU W., LIU K., SUN Z.: Model-driven indoor scenes modeling from a single image. In *Proceedings of the 41st Graphics Interface Conference* (2015), Canadian Information Processing Society, pp. 25–32. 2

[MRA15] MASSA F., RUSSELL B., AUBRY M.: Deep exemplar 2d-3d detection by adapting from real to rendered views. *arXiv preprint arXiv:1512.02497* (2015). 2

[NHH15] NOH H., HONG S., HAN B.: Learning deconvolution network for semantic segmentation. In *ICCV* (2015), pp. 1520–1528. 2, 4, 5, 6

[NSF12] NATHAN SILBERMAN DEREK HOIEM P. K., FERGUS R.: Indoor segmentation and support inference from rgbd images. In *ECCV* (2012). 7

[PGM*95] PICARD R., GRACZYK C., MANN S., WACHMAN J., PICARD L., CAMPBELL L.: Vistex texture dataset. http://vismod.media.mit.edu/vismod/imagery/VisionTexture/vistex.html, 1995. Accessed: 2016-05-01. 3, 5

[RHGS15] REN S., HE K., GIRSHICK R., SUN J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS* (2015), pp. 91–99. 7

[SLX15] SONG S., LICHTENBERG S. P., XIAO J.: Sun RGB-D: A RGB-D scene understanding benchmark suite. In *CVPR* (2015), pp. 567–576. 2, 3, 4, 7

[SMNS*13] SALAS-MORENO R. F., NEWCOMBE R. A., STRASDAT H., KELLY P. H. J., DAVISON A. J.: Slam++: Simultaneous localisation and mapping at the level of objects. In *CVPR* (2013), pp. 1352–1359.

[SMS15] SRIVASTAVA N., MANSIMOV E., SALAKHUTDINOV R.: Unsupervised learning of video representations using LSTMs. In *ICML* (2015).

[SQLG15] SU H., QI C. R., LI Y., GUIBAS L. J.: Render for CNN: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *ICCV* (2015), pp. 2686–2694. 2

[SSF14] SILBERMAN N., SONTAG D., FERGUS R.: *Instance Segmentation of Indoor Scenes Using a Coverage Loss*. 2014, pp. 616–631. 2

[SX14] SONG S., XIAO J.: Sliding shapes for 3d object detection in depth images. In *ECCV*. Springer, 2014, pp. 634–651. 2

[SZ14] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. *ICLR* (2014). 2, 3, 5

[TBF05] THRUN S., BURGARD W., FOX D.: *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. 3

[TH12] TIELEMAN T., HINTON G.: Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012. 4

[WHC*15] WEI L., HUANG Q., CEYLAN D., VOUGA E., LI H.: Dense human body correspondences using convolutional networks. *arXiv preprint arXiv:1511.05904* (2015). 2

[Wil10] WILSON S.: Dwelling Size Survey, 2010. URL: http://goo.gl/787jUF. 3

[WSK*15] WU Z., SONG S., KHOSLA A., YU F., ZHANG L., TANG X., XIAO J.: 3d shapenets: A deep representation for volumetric shapes. In *CVPR* (2015), pp. 1912–1920. 2

[ZJRP*15] ZHENG S., JAYASUMANA S., ROMERA-PAREDES B., VINEET V., SU Z., DU D., HUANG C., TORR P. H.: Conditional random fields as recurrent neural networks. In *ICCV* (2015), pp. 1529–1537.

[ZZ13] ZHAO Y., ZHU S.-C.: Scene parsing by integrating function, geometry and appearance models. In *CVPR* (2013), pp. 3119–3126. 2