Self-Supervised Real-to-Sim Scene Generation

Aayush Prakash Shoubhik Debnath Jean-Francois Lafleche
Eric Cameracci Gavriel State Stan Birchfield Marc T. Law
NVIDIA

Abstract

Synthetic data is emerging as a promising solution to the scalability issue of supervised deep learning, especially when real data are difficult to acquire or hard to annotate. Synthetic data generation, however, can itself be prohibitively expensive when domain experts have to manually and painstakingly oversee the process. Moreover, neural networks trained on synthetic data often do not perform well on real data because of the domain gap. To solve these challenges, we propose Sim2SG, a self-supervised automatic scene generation technique for matching the distribution of real data. Importantly, Sim2SG does not require supervision from the real-world dataset, thus making it applicable in situations for which such annotations are difficult to obtain. Sim2SG is designed to bridge both the content and appearance gaps, by matching the content of real data, and by matching the features in the source and target domains. We select scene graph (SG) generation as the downstream task, due to the limited availability of labeled datasets. Experiments demonstrate significant improvements over leading baselines in reducing the domain gap both qualitatively and quantitatively, on several synthetic datasets as well as the real-world KITTI dataset.

1. Introduction

Synthetic data, for which annotations can be automatically generated, is a promising solution to overcome the well-known supervised learning bottleneck of labeling real data. For this approach to succeed, such synthetic data must look like real data, in both appearance and content. Differences in appearance and content together comprise the so-called “domain gap” between synthetic and real data [55, 27]. Appearance refers to aspects like color, texture, shape, and lighting, whereas content refers to the number, position, and orientation of objects in the scene, as well as their relationships to one another.

At one extreme, synthetic scenes can be created manually by domain experts to reduce these gaps, but such solutions are expensive and therefore do not scale well [61, 47, 45, 18]. At the opposite extreme, domain randomization intentionally leverages these gaps to facilitate sim-to-real transfer [54, 41, 56]; these approaches, however, fail to capture the complexity and distribution of real scenes, thus fundamentally limiting their performance.

Recent approaches such as Meta-Sim [27], Meta-Sim2 [14], and SceneGen [52] attempt to reduce the content gap by automatically learning to generate synthetic data that matches the distribution of real data. However, Meta-Sim can only learn the position and orientation of objects, and therefore cannot align the
structure of the synthetic scenes (e.g., number and types of objects in a scene) to real data. Similarly, Meta-Sim2 only learns the distribution of one type of object (cars), and thus cannot match the surrounding context. Both Meta-Sim and Meta-Sim2 rely on a complex, realistic simulator designed on a set of hand-crafted heuristic rules to ensure that data are generated properly. SceneGen addresses these limitations but requires access to a large amount of labeled real data, which undermines the underlying purpose of synthetic data. None of these approaches addresses the appearance gap.

In this paper, we address both the content and appearance gaps, and we do so in a self-supervised manner that requires no real-world labels. Given an unlabeled real dataset, our method aims to automatically generate synthetic data that matches the distribution of the real data. See Figure 1. We propose Sim2SG (Simulation to Scene Graph), a synthesis-by-analysis framework, that generates scenes via self-learning [72] loop comprising of two alternating stages: synthesis and analysis. During the synthesis stage we leverage a synthetic data generator to create scenes. By ensuring that the number of objects, as well as their type and placement, are similar, the synthesized data resembles the real data, and therefore the content gap is reduced. During the analysis stage, we use the generated synthetic data for training. To further reduce both the content and appearance gaps, the corresponding latent and output distributions are aligned via Gradient Reversal Layers [19].

We show the effectiveness of our method in the scene graph generation task [11, 65, 66]. A scene graph (SG) summarizes entities in the scene and plausible relationships among them. The difficulty of hand-labeling scene graphs has limited the community to a small number of datasets [36, 29]. We experimentally demonstrate our method in three distinct environments: synthetic CLEVR [25], synthetic Dining-Sim, and real KITTI [20]. We nearly close the domain gap in the CLEVR environment and show significant improvements over respective baselines in Dining-Sim and KITTI. Through ablations, we validate our contributions regarding appearance and content gaps.

Contributions: Our contributions are three-fold: (1) To the best of our knowledge, we are first to do self-supervised aligned scene generation. (2) We propose a novel synthesis-by-analysis framework that addresses both the content and appearance gaps without using any real labels. (3) Experimentally, we show that Sim2SG obtains significant improvements on downstream tasks over baselines in all three scenarios: CLEVR, Dining-Sim, and KITTI. We also present ablations to illustrate the effectiveness of our technique.

2. Related Work

Synthetic Data has been used for many tasks including, but not limited to, object detection [27, 41, 56], semantic segmentation [45, 47, 58], optical flow modeling [4, 16], scene flow [38], classification [1, 3], stereo [43, 67], 3D keypoint extraction [50], object pose estimation [40, 57] and 3D reconstruction [39]. There are also several simulators [15, 12, 28, 53, 10, 13, 63] available for generating synthetic data. However, to the best of our knowledge, synthetic data has not been applied to scene graph generation.

Domain Gap is the performance gap when the network is trained on a synthetic source domain and evaluated on real target data. Most prior work addresses the appearance gap by image translations [8, 17, 23, 24, 32, 71], clever feature alignment [7, 34, 37, 48, 64, 30] and domain randomization [41, 54, 56]. There are few works which handle the content gap [2, 35, 51, 68] by addressing the label shift between the two domains. However, they do not exploit the unlabeled images from the target domain. We, on the other hand, leverage the images from the target domain to reduce the domain gap further. To this end, we exploit a scene graph generation task which is more complex than classification and relies on self-training [72]. The idea of self-training with pseudo labels is used in [34, 51] to learn models from the target distribution. However, the labels predicted by the model on the target are often inaccurate because of the domain gap [69]. We instead propose to rely on a synthetic data generator to produce accurate labels. Some approaches [6, 49] similar to ours also train their task model on top of domain invariant features for image classification and image segmentation. However, they do not address the content gap.

Scene Generation has been used in several machine learning driven approaches. For instance, some methods propose ways to learn how to synthesize indoor scenes [59, 70, 42, 31]. LayoutVAE [26] generates scenes conditioned on label sets. Deep priors and models [60, 46] synthesize scenes by sequentially placing objects. A few techniques [70, 59] also represent scenes as scene graphs with relationships among objects in the same way as our approach. SceneGen [52] learns to generate traffic scenes by modeling the attributes of all objects. However, all these methods either do not address the content gap with real data or use labeled real data for supervision. Meta-Sim [27] learns the distribution of position and rotation of objects in the scene and also requires small labeled real data. Meta-Sim2 [14] additionally learns the distribution of number and type of objects in the scene without needing labeled real data. However, it does not match the distribution of context. Both Meta-Sim and Meta-Sim2 rely on a
3. Reducing the domain gap

Let \( \langle x_r, y_r \rangle \sim p(x, y) \) be the real data and labels (where the labels \( y_r \) are not known). Our goal is to generate synthetic data \( \langle x_s, y_s \rangle \sim p(x, y) \) such that the distributions \( p \) and \( q \) match (i.e., \( p \approx q \)). We assume that both real and synthetic domains share the same categories of objects as well as scenario (e.g., both are driving scenes). The difference between \( p \) and \( q \) is the domain gap.

We now study this domain gap between synthetic and real domains. Let \( \phi \) be a network that encodes an input image \( x \) into a latent representation \( z \in \mathcal{Z} \), and let \( h \) be a network that estimates some desired quantity from \( z \). (In our case, \( h \) infers a scene graph.) We consider as in [62] that the task error in the real domain, \( \epsilon_r(\phi, h) \), is a function of three terms:

\[
\epsilon_r(\phi, h) = \int q(z) \epsilon_r dz
= \left[ \int p(z) e_s dz + \int q(z) (\epsilon_r - e_s) dz + \int (q(z) - p(z)) e_s dz \right]_{\epsilon_s(\phi, h)}^{\epsilon_r(\phi, h)}
\]  

(1)

where \( e_r \equiv |h(\phi(x_r)) - y_r| \) is the real risk, and similarly \( e_s \equiv |h(\phi(x_s)) - y_s| \) is the synthetic risk; and where the distributions of shared features for synthetic and real domains are denoted by \( p(z) \) and \( q(z) \), respectively. In this equation, \( e_s(\phi, h) \) is the training error on the synthetic domain, \( \epsilon' \) is the risk gap between the domains, and \( e^\phi(\phi, h) \) is the feature gap between the two domains. In the following, we drop the terms \( \phi, h \) for notational simplicity.

If the error \( \epsilon_r \) reduces to zero on the target domain, we have closed the domain gap. To reduce this error, we must reduce both \( \epsilon^c \) and \( \epsilon^a \). We call the former the content gap because it refers to the difference in the distributions of the two domains regarding the number of objects and their class, placement, pose and scale. The difficulty of minimizing the content gap is that computing it is intractable since we do not have access to labels \( y_r \) from the real domain, and hence \( \epsilon^c \) cannot be computed. We call the gap \( \epsilon^a \) the appearance gap because it refers to differences in texture, color, lighting, and reflectance of the scene. Recent work [21] also shows that latent representations \( z \) frequently account for appearance. In the following, we describe our approach to reduce both the content and appearance gaps.

4. Simulation to Scene Graph (Sim2SG)

Figure 2 illustrates our proposed Sim2SG pipeline, which comprises two alternating steps. During synthesis (real-to-sim), synthetic data is generated to match the distribution of the unlabeled real data. During analysis (sim-to-real), a scene graph (SG) prediction network [65] is trained using the ground truth labels of the synthetic data. This is analogous to Expectation-Maximization, where synthesis is like the E-step, and analysis is like the M-step. Results from different iterations of this self-learning loop, which is initialized using synthetic data generated by structured domain randomization (SDR) [41], are shown in Figure 3.

We now describe the synthesis and analysis steps. Algorithm 1 illustrates the pseudocode of both steps.

### 4.1. Synthesis Step

In this step, we aim to generate synthetic data that matches the distribution of unlabeled real data, thereby reducing the content gap. As mentioned previously, minimizing \( \epsilon^c \) in the current form is not tractable. Nevertheless, a sufficient condition for \( \epsilon^c \) to be zero is for \( y_s = y_r \) and \( h(\phi(x_r)) = h(\phi(x_s)) \). This leads to splitting the content gap into two terms:

\[
\epsilon^c = \int q(z)(\epsilon_r - e_s) dz 
\approx \left[ \int q(z)(y_s - y_r) dz + \int q(z)(h(\phi(x_r)) - h(\phi(x_s))) dz \right]_{e^c, \text{label}}^{e^c, \text{pred}}
\]  

(2)

where \( e^c, \text{label} \) refers to the gap between the ground truth labels of synthetic and real data, and where \( e^c, \text{pred} \) refers to the gap between the output of the network \( h \) trained on synthetic and real data.

Since we do not have access to the labels \( y_r \) of the target (real) domain, we propose to estimate \( y_r \) through...
Figure 2. Overview of Sim2SG, a self-supervised synthesis-by-analysis framework that generates synthetic data in a loop comprising of two alternating stages: synthesis and analysis. During synthesis (real-to-sim), we infer scene graphs from real data to generate synthetic data that matches the distribution of real data, thus bridging the content label gap, $\epsilon_{c,\text{label}}$. During analysis (sim-to-real), we train a scene graph prediction network ($\phi, h$) using the synthetic data. Additionally, Gradient Reversal Layers (GRLs) are used to align the latent features $z$ and output features $h(z)$ to bridge the appearance $\epsilon_a$ and content prediction $\epsilon_{c,\text{pred}}$ gap, respectively.

Figure 3. As the self-learning loop progresses, synthetic data increasingly matches the content of the target image. From left to right: synthetic data initialized by SDR [41], after first and third epochs, and corresponding real KITTI target images.

a self-supervised method that infer scene graphs from the target data. Our scene graphs consist of nodes and edges. Each node $o_i = \langle b_i, c_i \rangle$ consists of a bounding box $b_i = \{u_i, v_i, w_i, h_i\}$ (center, width, and height of the 2D box) and category $c_i$ (such as car, pedestrian, building, etc.). Each edge captures a relationship $r_i = \langle o_i, p, o_j \rangle$ where $p$ is a predicate (such as behind, left, on, etc.).

Using known camera parameters and a flat ground plane assumption, the scene graph can be mapped to a full 3D scene. Some parameters (e.g., texture or pose) and some context (ground, sky, lighting), that are not part of the scene graph are randomized. One advantage of generating synthetic data is that we can augment the relationships of the inferred scene graph by adding new relationships, simply by reasoning in the 3D representation. A synthetic data generator is then used to render the 3D scenes. Note that the weights of our networks ($\phi, h$) remain unchanged during this step.

4.2. Analysis Step

In this step, we train the encoder $\phi$ and predictor $h$ of our scene graph (SG) generation model on the synthetic data generated by the preceding step. The task loss function is from Yang et al. [65], which includes cross entropy loss for object classification and relationship classification, and $\ell_1$ loss for bounding boxes.

Although using synthetic data aligns the content label gap $\epsilon_{c,\text{label}}$, it does not align the appearance gap $\epsilon_a$ nor content prediction gap $\epsilon_{c,\text{pred}}$. To address the appearance gap, we align the feature distributions $p(z)$ and $q(z)$, since $p(z) = q(z)$ is a sufficient condition for $\epsilon_a$ to be zero [62]. To do this, during training $z$ is passed through a Gradient Reversal Layer (GRL) [19, 7], followed by a domain classifier $D^a$ that learns to classify the input as synthetic or real. The GRL acts as an identity function during forward propagation, but flips the sign of the gradients during back propagation. The
classifier $D^s$ provides an additional loss term for training $\phi$. We minimize $D^s$’s loss w.r.t. its own parameters while maximizing w.r.t. the network parameters of $\phi$. The loss function for training is as follows:

$$
L^a = - \sum_x \left[ d_i \log D^s(\phi(x)) + (1-d_i) \log(1-D^s(\phi(x))) \right]
$$

(3)

where $x \in x_s, x_r$, $d_i = 0$ for synthetic images $x_s$ and $d_i = 1$ for real images $x_r$.

Similarly, we seek to minimize the content prediction gap $\epsilon^{c,pred}$ by matching the distributions of the outputs $h(\phi(x_s))$ and $h(\phi(x_r))$ using the same GRL-based technique. This is based on the observation that the output of the scene graph generation model should be similar for corresponding inputs from the synthetic and real domains. During training, the output $h(z)$ is passed through a GRL, followed by a domain classifier $D^c$. The classifier provides an additional loss term for training both $\phi$ and $h$. The loss function is computed as:

$$
L^c = - \sum_z \left[ d_i \log D^c(h(z)) + (1-d_i) \log(1-D^c(h(z))) \right]
$$

(4)

where $z \in \phi(x_s), \phi(x_r)$. Fully matching the source and target appearance distributions without regard to content may cause detrimental results [48, 62]. We introduce a warm-up period during which content is aligned without regard to appearance or content prediction.

5. Experiments

We evaluate Sim2SG in three different environments with increasing complexity: CLEVR [25], our own Dining-Sim using ShapeNet [5], and KITTI [20]. In each environment we have a fully labeled synthetic domain and unlabeled target domain with labeled test data. We use scene graph (SG) generation as the downstream task and implement the encoder $\phi$ using ResNet-101 [22] and the predictor $h$ using Graph R-CNN [65].

Quantitative evaluation metrics include detection mAP (mean average precision) @0.5 IoU (Intersection over Union in 2D) and relationship triplet recall @20 or @50 [29]. Note that relationship triplet recall implicitly includes object detection recall. All the mean and standard deviations are based on five runs, and all results are reported after saturation. Details of the environments, training parameters, and hyperparameters are in the appendix.

For notational simplicity, we use $\sigma^{c,label}$ to refer to our synthesis step (whose purpose is to reduce the content label gap $\epsilon^{c,label}$), $\sigma^a$ to refer to the first GRL (whose purpose is to reduce the appearance gap $\epsilon^a$), and $\sigma^{c,pred}$ to refer to the second GRL (whose purpose is to reduce the content prediction gap $\epsilon^{c,pred}$). Similarly, we use $\sigma^c$ to refer to the combination of $\sigma^{c,label}$ and $\sigma^{c,pred}$, whose collective purpose is to reduce the content gap.

5.1. Experiments on the CLEVR dataset

The goal of the experiments on the CLEVR environment [25] is to show that Sim2SG can address the domain gap in a simple controlled environment. There are three classes of objects: cube, sphere and cylinder; and four relationships: front, behind, left and right. The source and target domains were generated using disjoint object colors, object materials, margin between objects, and number of objects, to ensure a significant domain gap. Furthermore, the target images have more complex texture via application of a style transfer network. We use 1000 labeled images for source domain, 1000 unlabeled images for target domain for training, and 200 labeled source and target images for evaluation.

Results: Quantitative evaluation of Sim2SG is reported in Table 1. Compared with the baseline using SDR [41], our techniques for label alignment $\sigma^{c,label}$ and appearance alignment $\sigma^a$ drastically reduce the domain gap. The best results are achieved by combining these two ideas. With such a simple environment, however, we found $\sigma^{c,pred}$ to have negligible effect, and therefore it is not reported. Qualitative results of scene graph recall are shown in first row of Figure 4.

Ablations: We conduct two ablation experiments on the CLEVR dataset to confirm that appearance alignment $\sigma^a$ and label alignment $\sigma^{c,label}$ work as intended. In the first experiment, the source and target domains are created with the same color and texture but different number and placement of objects. Thus, there is a content gap, but no appearance gap. We observe that label alignment $\sigma^{c,label}$ closes the domain gap from 0.76 to 0.996 for Recall@20, whereas $\sigma^a$ leads to performance degradation. In the second experiment, the source and target domains have the same number of objects and placement but use different color and texture. Thus, there is an appearance gap, but no content gap. We observe that appearance alignment $\sigma^a$ reduces the domain gap, achieving 0.938 versus 0.339 for Recall@20 for the baseline, whereas label alignment $\sigma^{c,label}$ fails to have significant improvement on rela-

<table>
<thead>
<tr>
<th>Method</th>
<th>mAP@0.5 IoU</th>
<th>Recall@20</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDR [41]</td>
<td>0.723 ± 0.053</td>
<td>0.356 ± 0.047</td>
</tr>
<tr>
<td>Ours ($\sigma^{c,label}$)</td>
<td>0.832 ± 0.046</td>
<td>0.493 ± 0.064</td>
</tr>
<tr>
<td>Ours ($\sigma^a$)</td>
<td>0.821 ± 0.048</td>
<td>0.815 ± 0.026</td>
</tr>
<tr>
<td>Ours ($\sigma^a$, $\sigma^{c,label}$)</td>
<td>0.892 ± 0.024</td>
<td>0.888 ± 0.018</td>
</tr>
</tbody>
</table>

Table 1. Results of Sim2SG on the CLEVR target domain. Aligning both appearance and content yields the best results.
Figure 4. Qualitative results of Sim2SG on the target domain for CLEVR (top row), Dining Sim (middle row) and KITTI environments (bottom row). Objects are color coded. For better visibility, we only show partial scene graph for KITTI.

Table 2. Results of Sim2SG on the Dining-Sim target domain. In the last row, $\sigma^c$ includes both $\sigma^{c,\text{label}}$ and $\sigma^{c,\text{pred}}$.

<table>
<thead>
<tr>
<th>Method</th>
<th>mAP@0.5 IoU</th>
<th>Recall@50</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDR [41]</td>
<td>0.584 ±0.049</td>
<td>0.331 ±0.064</td>
</tr>
<tr>
<td>Ours ($\sigma^{c,\text{label}}$)</td>
<td>0.713 ±0.038</td>
<td>0.501 ±0.044</td>
</tr>
<tr>
<td>Ours ($\sigma^c, \sigma^a$)</td>
<td><strong>0.729 ±0.015</strong></td>
<td><strong>0.547 ±0.015</strong></td>
</tr>
</tbody>
</table>

5.2. Experiments on Dining-Sim

We created a dataset that we call Dining-Sim by placing objects created from ShapeNet objects [5] in realistic arrangements, with more complex textures and realistic lighting. This dataset has three classes of objects: chair, table and laptop; and there are five relationships: front, behind, left, right, and on. Quantitative results, shown in Table 2, agree with the findings of the previous section. Since this data is more complex, the warm-up period mentioned earlier is necessary, and therefore the label alignment $\sigma^{c,\text{label}}$ must be performed first. This label alignment drastically improves performance on the target domain, and the combination of label alignment $\sigma^{c,\text{label}}$, appearance alignment $\sigma^a$, and prediction alignment $\sigma^{c,\text{pred}}$ achieve the best results. For reference, the oracle performance on the target domain is 0.904 mAP@0.5 IoU and 0.846 Recall@50. Qualitative results are illustrated in the second row of Figure 4.

5.3. Real-world experiments on KITTI

In this section, we validate our approach on the real-world KITTI dataset [20]. For the synthetic domain, we implemented from scratch a simplified version of SDR [41], with a fixed camera, a subset of object classes, no post processing, and without curved road splines, all for faster data generation. The number of lanes, sidewalks, and various objects, along with their positions, pose, color, texture, as well as lighting settings are randomly selected. Note that, unlike SDR, heuristic rules are not used to avoid collisions or ensure realistic placement, which further demonstrates the power of our proposed method in automatically generating useful training data. We use four object classes: car, pedestrian, vegetation, house, and four relationships: front, left, right, behind. Relationships are constrained so that at least one node is a car, e.g.,
car behind car, vegetation left car, and so on. Although we have shown the ‘on’ relationship to work in the Dining-Sim environment, such relationships are trivial to predict in this driving environment because they are always true: cars are always on the road, pedestrians are on the sidewalk, and so forth. Therefore, we did not include them in the experiments.

A subset of the real KITTI data [20] from the 2D object detection suite are used as the target domain. We use 6000 unlabeled KITTI images for training, and 550 labeled KITTI images for evaluation. The latter images include not only the provided annotations for cars and pedestrians, but also annotations that we added for
vegetation and houses, along with relationships.  

For training, we used 6000 labeled synthetic images, and another 1000 labeled synthetic images for validation.

**Baselines:** We compare against methods for which code is publicly available. In particular, we compare against unsupervised scene-generation methods, including structured domain randomization (SDR) [41] and Meta-Sim [27]. We also compare against a popular self-learning method based on pseudo labels extracted by training directly on the real KITTI data after pre-training on SDR synthetic data [72]. Furthermore, we compare against unsupervised domain adaptation methods for object detection, which are based on aligning features from the source and target domains: DA Faster R-CNN [7], GPA [64], and SAPNet [30]. According to a recent survey paper [33], the latter two are state-of-the-art leading techniques. All baselines are trained on 6000 images from SDR, using the hyperparameters provided by the original authors.

**Results:** We evaluate these baselines, along with our method, on three KITTI evaluation modes for 2D object detection: easy, moderate and hard, which are based on object size, occlusion and truncation. For all three modes, our method yields significantly improved results over the baselines. Table 3 shows the object detection and scene graph generation results for KITTI hard; other results are in the appendix. The last three rows of the table show that most improvements come from label alignment $\sigma_{c,\text{label}}$ and appearance alignment $\sigma_a$. The combination of all three, $\sigma_{c,\text{label}}$, $\sigma_a$ and $\sigma_{c,\text{pred}}$, achieves the best results overall. Note that $\sigma_{c,\text{label}}$ alone is able to beat most baselines, thus demonstrating the efficacy of our synthesis step at automatically generating realistic training data without heuristics. Qualitative results are illustrated in the third row of Figure 4.

Insight as to why our method outperforms SDR [41] and Meta-Sim [27] can be gained by viewing the images generated by the various methods, shown in Figure 6. Our method generates synthetic data that better matches the distribution of the real data, because SDR lacks access to the object distributions in KITTI, and because Meta-Sim cannot align the structure of the scenes (e.g., the number of objects) and lacks the notion of relationships in its scene graphs. (Note that Meta-Sim2 [14] also lacks relationships, but code is not publicly available.) Furthermore, our method scales better with scene complexity compared with Meta-Sim, which requires passing expensive numerical gradients through a renderer. As a result, our training time is 12 hours on a single NVIDIA V100 GPU, compared with 72 hours for Meta-Sim. Figure 5 compares our method on the downstream task with SDR and Meta-Sim.

The reason our method outperforms domain adaptation techniques [7, 64, 30] is because feature alignment without content alignment is not effective, as we discussed briefly at the end of Section 3. Nevertheless, it may be possible to combine our label alignment technique $\sigma_{c,\text{label}}$ with domain adaptation methods, which we leave for future work. Our method always produces proper ground truth for the scene via synthetic data generator, even if the scene is not generated exactly according to the corresponding real image. Self-learning [72] methods based on pseudolabels from KITTI, on the other hand, can have errors in the ground truth due to incorrectly classified objects or imprecise bounding boxes, as explained in [69].
Table 3. Results on KITTI hard after training on labeled synthetic data and unlabeled real data. The class specific AP values for 2D object detection are reported at 0.5 IoU. The last column shows relationship triplet recall for scene graph generation.

<table>
<thead>
<tr>
<th>Method</th>
<th>Car</th>
<th>Pedestrian</th>
<th>House</th>
<th>Vegetation</th>
<th><a href="mailto:mAP@0.5IoU">mAP@0.5IoU</a></th>
<th>Recall@50</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDR [41]</td>
<td>0.382 ±0.029</td>
<td>0.168 ±0.017</td>
<td>0.211 ±0.023</td>
<td>0.174 ±0.010</td>
<td>0.234 ±0.006</td>
<td>0.070 ±0.007</td>
</tr>
<tr>
<td>Meta-Sim [27]</td>
<td>0.413 ±0.009</td>
<td>0.197 ±0.027</td>
<td>0.236 ±0.009</td>
<td>0.164 ±0.023</td>
<td>0.253 ±0.003</td>
<td>0.075 ±0.005</td>
</tr>
<tr>
<td>Self-learning [72]</td>
<td>0.312 ±0.006</td>
<td>0.167 ±0.015</td>
<td>0.191 ±0.003</td>
<td>0.263 ±0.006</td>
<td>0.233 ±0.004</td>
<td>0.062 ±0.003</td>
</tr>
<tr>
<td>DA Faster R-CNN [7]</td>
<td>0.424 ±0.028</td>
<td>0.170 ±0.029</td>
<td>0.200 ±0.041</td>
<td>0.169 ±0.014</td>
<td>0.241 ±0.014</td>
<td>0.074 ±0.015</td>
</tr>
<tr>
<td>GPA [64]</td>
<td>0.174 ±0.040</td>
<td>0.011 ±0.016</td>
<td>0.106 ±0.031</td>
<td>0.059 ±0.027</td>
<td>0.087 ±0.020</td>
<td>0.015 ±0.005</td>
</tr>
<tr>
<td>SAPNet [30]</td>
<td>0.362 ±0.054</td>
<td>0.085 ±0.051</td>
<td>0.116 ±0.021</td>
<td>0.067 ±0.022</td>
<td>0.157 ±0.024</td>
<td>-</td>
</tr>
<tr>
<td>Ours (σ&lt;sup&gt;c&lt;/sup&gt;,&lt;sup&gt;label&lt;/sup&gt;)</td>
<td>0.410 ±0.009</td>
<td>0.262 ±0.025</td>
<td>0.240 ±0.010</td>
<td>0.229 ±0.036</td>
<td>0.285 ±0.003</td>
<td>0.104 ±0.006</td>
</tr>
<tr>
<td>Ours (σ&lt;sup&gt;c&lt;/sup&gt;,&lt;sup&gt;label&lt;/sup&gt;, σ&lt;sup&gt;a&lt;/sup&gt;)</td>
<td>0.495 ±0.004</td>
<td>0.252 ±0.014</td>
<td>0.247 ±0.012</td>
<td>0.253 ±0.020</td>
<td>0.311 ±0.311</td>
<td>0.127 ±0.004</td>
</tr>
<tr>
<td>Ours (σ&lt;sup&gt;c&lt;/sup&gt;,&lt;sup&gt;label&lt;/sup&gt;, σ&lt;sup&gt;a&lt;/sup&gt;, σ&lt;sup&gt;c&lt;/sup&gt;,&lt;sup&gt;pred&lt;/sup&gt;)</td>
<td>0.501 ±0.006</td>
<td>0.241 ±0.018</td>
<td>0.254 ±0.010</td>
<td>0.269 ±0.014</td>
<td>0.316 ±0.004</td>
<td>0.139 ±0.004</td>
</tr>
</tbody>
</table>

Table 3: Results on KITTI hard after training on labeled synthetic data and unlabeled real data. The class specific AP values for 2D object detection are reported at 0.5 IoU. The last column shows relationship triplet recall for scene graph generation.

Note in Table 3 that the detection accuracy of the pedestrian category does not improve with σ<sup>a</sup> and σ<sup>c</sup>,<sup>pred</sup>. The reason for this limitation is that pedestrians are an under-represented class in KITTI, not to mention small and hard to detect. While our Sim2SG method can align the label distribution, it cannot address class imbalance in the target domain. Nevertheless, our method is not restricted to the types or number of relations. Given a simulator to generate scenarios such that objects and their relations are detectable, our method should extend to handle them.

Ablations: As briefly discussed at the end of Section 3, we run the label alignment σ<sup>c</sup>,<sup>label</sup> before appearance alignment σ<sup>a</sup> and prediction alignment σ<sup>c</sup>,<sup>pred</sup> to address the fact that feature alignment can be detrimental if the content of both domains are not already aligned. We indeed found that performance drops significantly when we train Sim2SG without σ<sup>c</sup>,<sup>label</sup> and evaluate in the same setting as Table 3. Sim2SG with σ<sup>a</sup> and σ<sup>c</sup>,<sup>pred</sup> gives a 0.246 mAP@0.5 IoU for detection and 0.076 Recall@50 for relationship triplets, while adding σ<sup>c</sup>,<sup>label</sup> yields a significant boost of 0.316 mAP@0.5 IoU for detection and 0.139 Recall@50 for relationship triplets on KITTI Hard. These results confirm the effectiveness of σ<sup>c</sup>,<sup>label</sup> and therefore the importance of the entire Sim2SG framework.

6. Conclusion

In this work, we have proposed Sim2SG, a self-supervised real-to-sim automatic scene generation technique that matches the distribution of real data, for the purpose of training a network to infer scene graphs. The method bridges both the content and appearance gap with real data without requiring any costly supervision on the real-world dataset. The approach achieves significant improvements over baselines in all three environments for which we tested: CLEVR, a Dining-Sim dataset, and real KITTI data. The latter of these demonstrates the ability of our method to perform sim-to-real transfer.
References


[22] Xin Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In ECCV, 2018. 2


[24] Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. LayoutVAE: Stochastic scene layout generation from a label set. In IEEE/CVF International Conference on Computer Vision (ICCV), 2019. 2


