Holistic static and animated 3D scene generation from diverse text descriptions

Faria Huq  
Bangladesh University of Engineering & Technology

Anindya Iqbal  
Bangladesh University of Engineering & Technology

Nafees Ahmed  
Waymo

Abstract
We propose a framework for holistic static and animated 3D scene generation from diverse text descriptions. Prior works of scene generation rely on static rule-based entity extraction from natural language description. However, this limits the usability of a practical solution. To overcome this limitation, we use one of state-of-the-art architecture - TransformerXL. Instead of rule-based extraction, our framework leverages the rich contextual encoding which allows us to process a larger range (diverse) of possible natural language descriptions. We empirically show how our proposed mechanism generalizes even on novel combinations of object-features during inference. We also show how our framework can jointly generate static and animated 3D scene efficiently. We modify CLEVR to generate a large, scalable dataset - Integrated static and animated 3D scene (Iscene). Data preparation code and pre-trained model available at - https://github.com/oaishi/3DScene_from_text.

1 Introduction
Rendering refers to the process which converts source materials into a realistic scene. In production, the characters, textures and animations are generated first, the layout is then set up with these predefined materials to render the final scene on a renderer (Burley et al., 2018; Kerlow, 2009). Although there is no alternative to human intervention for modelling high-quality materials, many functionalities are repetitive for each story plot- for example, the action of the characters (i.e, walk, jump, laugh etc). A system that can take text description and automatically generate the scene layout for rendering could improve accessibility substantially. The use case of such a system also advances to other domains such as gaming, simulation, interior design etc.

The concept of scene generation from natural language description was first introduced by (Adorni and Di Manzo, 1983). (Coyne and Sproat, 2001) first developed a system that takes a text description and renders a scene. However, the input text description needs to follow a certain pattern otherwise the rendered scene might not be consistent and meaningful. Later, the study has been more advanced by the advancement in machine learning (Chen and Manning, 2014; Goodfellow et al., 2014; Kipf and Welling, 2016). Many studies successfully generate 3D indoor scene (Chang et al., 2014, 2015, 2017) and interaction with objects (Balint et al., 2015; Krishnaswamy and Pustejovsky, 2016). However, the prior studies are often restricted to a static rule-based entity and relationship extraction which limits the scope of usability. As the choice of words and syntax varies from person-to-person, a practical solution needs to be able to process diverse text descriptions. Furthermore, the scope of prior works has been limited to either static scene or interaction between objects. In this study, we focus on the holistic generation of static and animated scene from diverse text descriptions. For this task, we prepare a wider range of description families than reported before, which are created to cover various types of possible descriptions of a scene. We deploy one of the state-of-the-art neural models, TransformerXL (Dai et al., 2019) for mapping the description to the abstract scene layout. To the best of our knowledge, this is the first study to deploy Transformer based module for 3D scene generation. We design a Blender program, referred to as script in the literature (Blender, 2020) which sets up and renders scenes with the extracted layout from our neural model. Overall, we focus on the following research questions in this study:

**RQ1:** Can we process a larger range of possible natural language scene descriptions without...
depending on rule-based entity and relation extraction? Can we leverage the deep context encoding of TransformerXL for this task? Will TransformerXL generalize well in all scenario?

**RQ2:** Can we jointly generate static and animated 3D scene using the same framework?

We provide an empirical study of our proposed framework in support of these research questions. Our contributions are as follows -

- We propose a novel two-stage framework for holistic static and animated 3D scene generation from diverse text descriptions. Our proposed framework leverages state-of-the-art architecture to process a wider range of possible natural language description.

- We generate Integrated Static and Animated Scene Generation (IScene) dataset. IScene is generated programmatically under a physics-based render engine which can be further extended and scaled easily.

- We provide empirical analysis of how our framework captures the natural language contexts efficiently and generates both static and animated scenes efficiently. We also demonstrate the ability of our approach to control scene objects even during inference.

## 2 Related Work

In this section, we overview previous works on related tasks.

### 2.1 3D scene synthesis from text description

The concept of scene generation from natural language description was first introduced in 1983 (Adorni and Di Manzo, 1983). Words-eye (Coyne and Sproat, 2001) first implemented a pipeline which takes natural language and renders a static scene. However, their generated scene might not always be coherent. (Seversky and Yin, 2006) designed a system which allows users to generate the final scene in an interactive manner from input voice and text description, where user can choose the input scene objects. (Chang et al., 2014, 2015) maps the input text description into a dependency graph (Chen and Manning, 2014) and learns spatial knowledge. The dependency graph is used to find out relative positioning of the scene objects. (Chang et al., 2017) allows to edit the scene and view from a specific position using natural language description and given input scene. (Savva et al., 2016; Zhao et al., 2016) learns interaction with daily life objects from real-world examples and generates these interactions from given input description. Similarly, (Krishnaswamy and Pustejovsky, 2016; Balint et al., 2015) generates animated scenes on a corpus of a limited amount of motions and objects. (Ma et al., 2018) can edit an input scene as per text instruction. They learn pair and group-wise positioning and existence of scene objects for coherence.

### 2.2 3D scene synthesis from Layout

(Merrell et al., 2011; Fisher et al., 2012) learns relative positioning from a scene database layout and synthesize a diverse set of plausible new scene layouts from a given input scene. (Fisher et al., 2015) provides functional plausibility of an arrangement of objects from an input 3D scan using an activity model learned from an annotated 3D scene and model database. (Zhou et al., 2019) proposes a neural message-passing approach to complete a given incomplete 3D scene and predict object type at a specific query location. (Wang et al., 2019) generates floor graph with object instances and positioning from input empty or incomplete graph. The graph is further used to instantiate a 3D scene by iterative insertion of 3D models. (Zhang et al., 2020) presents a fast framework for indoor scene synthesis, given a room geometry and a list of objects with learned priors. (Chen et al., 2020) proposes a novel 3D-house generation architecture. Their system takes linguistic description as input and uses Graph Convolutional Network to predict the room layout. Their system also includes a generative adversarial network module which generates floor textures of each room. The textures and layout are passed to the renderer to generate the final 3D house layout.

### 2.3 Image and video generation from text description

Over last decade, many studies proposed novel generative models for generating image from text description using Generative Adversarial Network (GAN) (Goodfellow et al., 2014) and Variational Autoencoder (Kingma and Welling, 2013). (Zhang et al., 2017; Xu et al., 2018; Reed et al., 2016; Chen et al., 2018; Yan et al., 2016; Mansimov et al., 2015; Li et al., 2019a) propose various kinds of novel architecture for generating images from text. (Johnson et al., 2018) generate images from scene graph. Some recent studies are focusing on image gener-
ation from text in an iterative manner where one or two actors engage in giving instruction about the scene (El-Nouby et al., 2019; Kim et al., 2017; Cheng et al., 2018; Benmalek et al., 2018; Tan et al., 2019; Sharma et al., 2018).

(Li et al., 2019b) generates a series of images which form a story altogether. A related study is to generate video from story plot (Gupta et al., 2018). (Pan et al., 2017; Li et al., 2018) generate videos from caption.

To the best of our knowledge, this is the first study on the inclusion of Transformer for scene generation from diverse text descriptions. Furthermore, we show how our pipeline generates both static and animated scenes which have not been studied before as our best knowledge.

3 Our Objective

For our task of joint generation of static and animated scene from diverse text descriptions, we focus on the dataset - CLEVR (A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning) (Johnson et al., 2017), extensively used in the literature. CLEVR is generated in the Blender Engine (Blender, 2020), using random combinations of primitives (Sphere, Cube and Cylinder) and object-features (color, shape, size and texture). CLEVR also provides Compositional Generalization Test (CoGenT), where the dataset is divided under two conditions (conditionA, conditionB). Their homepage provides more details on the dataset. We modify CLEVR to generate our Integrated Static and Animated Scene Generation (IScene) dataset.

We work on CLEVR for the following reasons. First, the object types, object features, location and camera position of each associated scene used in CLEVR are publicly available. Hence, we can use this information as ground truth to validate our approach. Second, CLEVR use question template for visual reasoning. These question templates use the ground truth information for generating diverse types of questions. Similarly, we can design description template for generating diverse types of descriptions. Third, each object has a minimum four associated features which provides a wide range of possible combinations and scene description. A quantitative description is a high-level abstraction of a scene where only the number of objects present in the scene are included. In figure 2, we can see

We will validate our approach from the following perspective:

1. Ability of our approach on similar data patterns: We will test our trained model on a similar combination of object features seen during training (referred to as conditionA/condA in CLEVR). The result will show how our framework captures known data patterns. We will use non-overlapping data points to train, test and validate our model.

2. Ability of our approach on novel data patterns: We want to test how our model performs on a novel combination of object features during inference (referred to as conditionB/condB in CLEVR). For example, any data points including red cube is not present in the training set. During inference, we will test our model on data points containing red cube. The result will show how our model adapts to new data patterns.

3. Editability of our approach: We want to test if our framework can provide us edit control. We will provide separate instances of the same scene objects to our renderer script and analyze if these are executed correctly.

For each of the aforementioned validation task, we consider both static and animated 3D scenes.

4 IScene dataset

Our framework will take text descriptions as input and generate the corresponding 3D scene. Hence, each data point in our dataset will comprise of natural language scene description and corresponding parameters of the 3D scene. For the current scope of our work, we consider three description families: Narrative, Semi-Narrative and Quantitative for our diverse text description. A narrative description contains a detailed description of the scene, information about each feature of each object type is mentioned. In figure 2, we can see each object-feature of the cube (color: yellow, size: large, texture: metal) is mentioned the narrative description. In practical cases, very detailed information of a scene might not always be available. To allow our framework to perform under the restricted scope of description, we consider semi-narrative descriptions. A semi-narrative will discard a randomly chosen feature about each object in a scene. In figure 2, we can see that size of the cube is not mentioned in the semi-narrative description. In figure 2, we can see

https://cs.stanford.edu/people/jcjohns/clevr/
that there are two spheres present in the scene and corresponding quantitative description. We use templates to generate the corresponding descriptions based on the ground truth information. These description templates can be extended further to generate various types of description.

**Narrative:** There is a large yellow shiny cube, a large brown shiny sphere, a large green matte cylinder and a large brown matte sphere.

**Semi-Narrative:** There is a yellow shiny cube, a large brown shiny sphere, a large matte cylinder and a large brown matte object.

**Quantitative:** There are two spheres, one cube and a cylinder in the scene.

Figure 2: An example static scene with corresponding description of each category

For an animated scene, we enhance ground truth information from CLEVR by inducing motion following (Girdhar and Ramanan, 2020). For the current scope of our work, we consider five simple atomic motions - Spin, Bounce, Shake, Move and Rest. We assign a motion with each scene object randomly. The effect of rest and spin are seemingly the same for cylinder and sphere, hence we do not consider spin for these two shapes in our data preparation. Figure 1d reports the distribution of the five motions in our dataset. Apart from spin, the distribution of the rest four motions are even. Due to the restriction over the motion - spin, it’s occurrence is lower than the rest. We generate motions as program templates which can be further extended to include a new variety of motions. The recorded animations are three seconds long in default. However, it can be tuned as well. The scene description for animated scenes is generated similar to the static scenes under three families as described above. An example animated scene from our dataset with the corresponding narrative description.

**Rest.** There is a large yellow shiny cube, a large brown shiny sphere, a large green matte cylinder and a large brown matte sphere.

**Semi-Narrative:** There is a yellow shiny cube, a large brown shiny sphere, a large matte cylinder and a large brown matte object.

**Quantitative:** There are two spheres, one cube and a cylinder in the scene.

Figure 3: An example animated scene with corresponding narrative description
Draw a cyan spinning large rubber cube, a red bouncing small metal sphere, a purple shaking large metal cylinder, a moving brown small cylinder and a still blue large metal sphere.

Figure 4: Our architecture. We first encode the scene description in an encoded vector using TransformerXL. Our novel decoder extracts the features of each objects. These abstract feature levels (abstract layout) are passed to the renderer to render which render the final static and/or animated scene.

The distribution of each object-feature in both static and animated scenes are even. Figure 1f reports the distribution of missing object-feature information in semi-narrative and qualitative description. It is worth mentioning, all plausible biases are carefully discarded by weighted loss calculation during training the model (Section 5.4).

5 Our Framework

In this section, we describe different components of our framework. Our framework has three major components: Description Encoder (Section 5.1), Hidden2ObjectFeature Decoder (Section 5.2) and Scene Renderer (Section 5.6). Figure 4 shows the overall architecture of our framework.

5.1 Description Encoder

To encode our scene description into a deep contextual vector, we use one of state-of-the-art architectures, TransformerXL (Dai et al., 2019). We use the pretrained model of TransformerXL from HuggingFace Distribution (Wolf et al., 2019). Transformers are proven to work well to capture the context and encode the information in a meaningful vector space. The motivation of including TransformerXL is to allow our framework capture the usefulness under low-resource constraints. The encoding is done according to the following equation:

\[ H_0 = E(T_d) \] (1)

Where \( T_d \) is the scene description containing \( d \) words, \( H_0 \in \mathbb{R}^{d \times 1024} \) represents the encoded hid-
We propose a novel decoder, Hid-
while generating a specific output token. For
hid2object feature extractors (Section 5.1), H
features (Section 5.1), previous hidden
state, \( h_{t-1} \). Attention mechanism allows us the
understand and capture the corresponding portions
of input description which are more important
while generating a specific output token. For
example, intuitively, the words “a cyan spinning
large rubber cube” are most important while
generating the cube as output in Figure 4. We will
further analyze the output of our attention layer in
Section 7.

After applying attention on \( H_0 \) and \( h_{t-1} \), we get
the context vector, \( \hat{c}_t \). The addition of \( \hat{c}_t \) and input,
\( i_t \) - is passed through the LSTM layer. Specifically,
\[
\begin{align*}
    a_t &= \tanh(W_i H_0 + W_o h_{t-1} + b_i) \quad (2) \\
    \hat{a}_t &= \text{softmax}(W_{\text{att}} a_t + b_2) \quad (3) \\
    \hat{c}_t &= \hat{a}_t H_0 \quad (4) \\
    y_t, h_t &= \text{LSTM}(\hat{c}_t + i_t, h_{t-1}) \quad (5)
\end{align*}
\]

Here \( W_i, b_i, h_n \) in the formula specify weight
matrices, biases and hidden vectors respectively, \( \hat{a}_t \) is
the attention distribution over input vectors \( H_0 \) for
generating the output \( y_t \) as time-step \( t \). The input
\( i_t \) is generated by the \( N \) embedding layers of the
decoder, in the previous time step \( t-1 \) (discussed
later in Section 5.3)

The output, \( y_t \) from LSTM is expected to be
representing one object in the scene. \( y_t \) is passed
through \( N \) separate dense layers to generate \( N \)
object-features, where \( N \) is the number of object-
features. The motivation is to predict each object-
feature separately, because they are mutually indepen-
dent. Let \( f_t = [f_{t,1}, f_{t,2}, \ldots, f_{t,N}] \) be \( N \) object-
features generated from \( y_t \). \( f_{t,i} \in f_t \) can be calculated using the Equation 6. This is depicted
as Object Feature Extractors in Figure 4.

\[
    f_{t,i} = \text{softmax}(W_{f_i} y_t + b_{f_i}) \quad (6)
\]

Here, \( W_{f_i} \) and \( b_{f_i} \) are the weight and bias of the
dense layer that generates the \( i-th \) object-feature
\( f_{t,i} \).

### 5.2 Hidden2ObjectFeature Decoder

We propose a novel decoder, Hidden2ObjectFeature which maps the encoder
hidden space to object-feature distribution. The extracted object-features create an abstract layout
which is then passed to the scene renderer to
generate 3D scene.

We use single-layer uni-directional
LSTM (Hochreiter and Schmidhuber, 1997)
for our base decoder unit. We apply attention
(Bahdanau et al., 2014) on the encoded hidden
vectors (Section 5.1), \( H_0 \) and previous hidden
state, \( h_{t-1} \). Attention mechanism allows us the
understand and capture the corresponding portions
of input description which are more important
while generating a specific output token. For
example, intuitively, the words “a cyan spinning
large rubber cube” are most important while
generating the cube as output in Figure 4. We will
further analyze the output of our attention layer in
Section 7.

After applying attention on \( H_0 \) and \( h_{t-1} \), we get
the context vector, \( \hat{c}_t \). The addition of \( \hat{c}_t \) and input,
\( i_t \) - is passed through the LSTM layer. Specifically,
\[
\begin{align*}
    a_t &= \tanh(W_i H_0 + W_o h_{t-1} + b_i) \quad (2) \\
    \hat{a}_t &= \text{softmax}(W_{\text{att}} a_t + b_2) \quad (3) \\
    \hat{c}_t &= \hat{a}_t H_0 \quad (4) \\
    y_t, h_t &= \text{LSTM}(\hat{c}_t + i_t, h_{t-1}) \quad (5)
\end{align*}
\]

Here \( W_i, b_i, h_n \) in the formula specify weight
matrices, biases and hidden vectors respectively, \( \hat{a}_t \) is
the attention distribution over input vectors \( H_0 \) for
generating the output \( y_t \) as time-step \( t \). The input
\( i_t \) is generated by the \( N \) embedding layers of the
decoder, in the previous time step \( t-1 \) (discussed
later in Section 5.3)

The output, \( y_t \) from LSTM is expected to be
representing one object in the scene. \( y_t \) is passed
through \( N \) separate dense layers to generate \( N \)
object-features, where \( N \) is the number of object-
features. The motivation is to predict each object-
feature separately, because they are mutually indepen-
dent. Let \( f_t = [f_{t,1}, f_{t,2}, \ldots, f_{t,N}] \) be \( N \) object-
features generated from \( y_t \). \( f_{t,i} \in f_t \) can be calculated using the Equation 6. This is depicted
as Object Feature Extractors in Figure 4.

\[
    f_{t,i} = \text{softmax}(W_{f_i} y_t + b_{f_i}) \quad (6)
\]

Here, \( W_{f_i} \) and \( b_{f_i} \) are the weight and bias of the
dense layer that generates the \( i-th \) object-feature
\( f_{t,i} \).

### 5.3 Multi Decoder-Embedding

To further ensure equal emphasis of each object-
feature, the input for the next state is obtained
from the object-features of the current state. For
each of the object-features, there is one unique embed-
ning layer \( \epsilon_i(\cdot) \), generating the object-feature-
embedding vector \( \text{emb}_{t,i} \) according to Equation 7.

\[
    \text{emb}_{t,i} = \epsilon_i(f_{t,i}) \quad (7)
\]

The embedding vectors \( [\text{emb}_{t,1}, \text{emb}_{t,2}, \ldots, \text{emb}_{t,N}] \) are concatenated
to generate the input vector \( i_{t+1} \), which is then summed with the context vector generated by the
attention mechanism (as shown in Equation 5).

### 5.4 Loss Function

Intuitively, each object-feature plays an equal role
for a successful scene generation. Hence, we cal-
culate weighted negative log likelihood loss\(^2\) for
each object-feature. The weighting in considered
to nullify any bias in the dataset.

For calculating the loss, the rescaling weight
given to each class is calculated according to the
distribution of object-features of our dataset. This
distribution is depicted in Figure 1.

### 5.5 Object-Feature Accuracy Metric

As each object-feature is equally important for
a successful scene generation, during evaluation
we check if each object-feature, \( f_{t,j} \) of each object
matches with the ground truth information, \( t_{i,j} \). So,
the Object-Feature Accuracy\(^3\), \( acc \), can be derived
using Equation 8 and 9.

\[
    acc = \frac{1}{l \times n} \sum_{i=1}^{l} \sum_{j=1}^{n} a_{i,j} \quad (8)
\]

\[
    a_{i,j} = \begin{cases} 
        0 & \text{if } f_{i,j} \neq t_{i,j} \\
        1 & \text{otherwise} 
    \end{cases} \quad (9)
\]

Here, \( l \) refer to number of objects in the scene
and \( n \) is the number of object-features for each of the
objects.

\(^2\) https://pytorch.org/docs/master/generated/torch.nn.NLLLoss.html

\(^3\) Object-Feature Accuracy Calculation code - https://
github.com/oaishi/3DScene_from_text/
blob/master/scripts/evaluation_metric.py
5.6 Scene Renderer

To render the final 3D scene from the abstract layout output of decoder, we prepare a Blender scene renderer script. Our script takes the layout and perform all necessary post-processing which includes camera setup, light positioning, 3D model loading and animation setup. An important functionality of scene renderer is to remove ambiguities such as collision among scene objects.

6 Experiment

In this section, we overview our experimental setup (Section 6.1), evaluation metrics (Section 6.2) and hyperparameter settings (Section 6.3).

6.1 Experimental Setup

To the best of our knowledge, there has not been any prior study on holistic static and animated 3D scene generation from diverse text descriptions. Hence, We conduct and evaluate our study three variations of our baseline model. Our first model, \( M_{\text{static}} \), is trained on static scene descriptions. Our second model, \( M_{\text{animated}} \), is trained on animated scene descriptions. Our third and final model is an oracle model, \( M_{\text{full}} \), that is trained on both static and animated scene descriptions. We prepare a separate train, validation and test set for each model. The overview of the data count used for each model configuration is used in Table 2. It is worth mentioning, each train, validation and test set contain non-overlapping data points extracted from the total dataset (Table 1).

| Model     | Train     | Validation | Test          |
|-----------|-----------|------------|---------------|
| \( M_{\text{static}} \) | 1,00,000 | condA: 5000 | condA: 6400 |
|           |           | condB: 5000 | condB: 6400   |
| \( M_{\text{animated}} \) | 1,00,000 | condA: 5000 | condA: 6400 |
|           |           | condB: 5000 | condB: 6400   |
| \( M_{\text{full}} \)   | \( a \times 50,000 \) | condA: 2500(a) + 2500(a) | condA: 3200(a) + 3200(a) |
|           | \( a \times 50,000 \) | condB: 2500(a) + 2500(a) | condB: 3200(a) + 3200(a) |

Table 2: Train-Validation-Test Split (\( s \) and \( a \) denote static and animated description respectively)

6.2 Evaluation Metric

We evaluate our framework on the following metrics.

6.2.1 Object-Feature Accuracy

As discussed in Section 5.5, we evaluate if each object-feature is the same as the ground truth object-feature information. Object-feature accuracy will show how accurately our decoder (Section 5.2) can capture the natural language description.

6.2.2 Structural Similarity Index (SSIM)

Structural Similarity Index (Wang et al., 2004) is a standard metric for predicting the perceived quality of images and videos (Li et al., 2019b). After the final scene is rendered (Section 5.6), we use SSIM to compare the generated image (static scene) and video (animated scene) with the ground truth. The result will show how accurately our framework can generate scenes with respect to ground truth as well as capture the natural language description.

6.3 Hyperparameter Settings

Throughout our experiments, we use the following hyperparameter settings. We train each of our models on NVIDIA GeForce GTX 1070 for 78 hours.

- Batch size, \( b = 520 \)
- Learning rate = 0.01
- Dropout Rate = 0.1
- Teacher Forcing Ratio = 50%
- LSTM Hidden State Dimension = \( b \times 1024 \) (following \( H_0 \) (Section 5.1))
- LSTM Input Dimension = \( b \times 1024 \)
- Dense Layer Input Dimension = \( b \times 1024 \)
- Dense Layer Output Dimension = \( b \times f_t \)

7 Result and Discussion

We train each of our models for around 120 epochs with a validation step after every 10 epochs. As Figure 6 reports, the expected performance stabilizes and the error decreases significantly after 120 epochs. At this point, We report our final accuracy over the metrics discussed in Section 6.2 in Table 3.

| Model     | Known combination | SSIM | Novel combination | SSIM  |
|-----------|--------------------|------|-------------------|-------|
| \( M_{\text{static}} \) | 98.427             | 0.801| 94.270            | 0.809 |
| \( M_{\text{animated}} \) | 97.482             | 0.906| 93.234            | 0.857 |
| \( M_{\text{full}} \)   | 94.910             | 0.812| 90.040            | 0.813 |
|           | \( s \): 0.849     |      |                   | 0.888 |

Table 3: Accuracy of our models on test set (\( s \) and \( a \) denote static and animated description respectively)
The results show that our framework performs consistently both for $M_{\text{static}}$, $M_{\text{animated}}$, $M_{\text{full}}$. The slight decrease in object-feature accuracy of $M_{\text{animated}}$ from $M_{\text{static}}$ is due to inclusion of motion which enforces the model to learn one extra object-feature (motion) than $M_{\text{static}}$. Similarly, the decrease in object-feature accuracy of $M_{\text{full}}$ is due to the extra constraints over the model. However, as we can see the relative difference between these configurations is very less. This proves the effectiveness of our proposed framework on both static and animated scene. The accuracy also shows that the model performs almost similarly on holistic (both static and animated scene) generation and disjoint (either static or animated scene) generation. This provides the answer to RQ2 (Section 1).

The performance of each model on known and novel combination is also noteworthy. The small margin of accuracy between these combinations shows our framework has a very small bias on data pattern. This provides the answer to RQ1 (Section 1). Overall, the higher accuracy strengthens the applicability and potential of our system for holistic scene generation from diverse text descriptions.

### 7.1 Case Study

In this section, we present and analyze two particular outputs from our framework in Figure 5.

On the left, we can see the generated and ground truth static scene with corresponding narrative scene description. As we can see, every object feature - color, shape, size and texture matches with the ground truth for all nine scene objects. However, the relative positioning is not the same as the ground truth.

On the right, we can see the generated and ground truth animated scene with corresponding semi-narrative scene description. As we can see, every motion of all the six objects matches with the ground truth and input description. However, ‘a small moving shiny cube’ has been mapped to a ‘a small blue moving shiny cube’ whereas ground truth scene is ‘a small yellow moving shiny cube’. Our framework successfully maps the texture, shape and size mentioned for this object. As the color of the cube was not mentioned in the input, it was impossible to infer the ground truth information, hence the generated animated scene is correct with respect to the given information.

---

[4] More example available at: https://github.com/oaishi/3DScene_from_text.
Figure 7: Attention map for a successful test sample generated by our framework. Scene description and predicted outputs are shown on X and Y axis respectively.

Figure 8: Our framework allows us to control and change the scene objects even during inference.

**7.2 Result Analysis**

In this section, we analyze if our model captures the context correctly. For this, we focus on the output of the attention layer discussed in Section 5.2. Figure 7 shows the attention map of a successful scene description during inference. As the value gets closer to 1 and as the color goes closer to white, the word is given more importance (higher attention weight) by our model. As expected, cube, red, color, shiny are more important while generating the first object - ‘cube’. Similarly, we can see the words - cylinder and sphere are more important while generating the second and third object - ‘cylinder’ and ‘sphere’ respectively. However, there is also some false positive values. For example, the last word texture’ is not much important for the second object - ‘cylinder’.

**7.3 Editability**

For our third task, we explain the finer control of our mechanism. Figure 8 reports a successful data-point in inference. Although the scene description of both scenes is the same, the first scene includes an icosphere which represents a sphere in low-poly graphics. The second scene includes the regular smooth sphere used in the literature. Thus, using the same description and same trained parameters, our framework allows us to control and change the settings, objects and features if needed, which is practically more useful.
8 Conclusion and Future Work

We introduce a novel task of holistic static and animated 3D scene generation from diverse text descriptions. We deploy state-of-the-art natural language architecture and observe a promising result. We further show various analysis and strengths of our framework. Our proposed method can jointly generate static and animated scene with a longer coverage of text description than reported before. In future, we wish to deploy advanced Graph Network modules, for more complex scene description including relational description to handle relative positioning.

References

Giovanni Adorni and Mauro Di Manzo. 1983. Natural language input for scene generation. In First Conference of the European Chapter of the Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

John Balint, Jan M Allbeck, and Michael R Hieb. 2015. Automated simulation creation from military operation documents. In Interservice/Industry Training, Simulation and Education Conference, volume 15227. I/ITSEC Orlando, Florida.

Ryan Y. Benmalek, Claire Cardie, Serge Belongie, Xiaodong He, and Jianfeng Gao. 2018. The neural painter: Multi-turn image generation.

Blender. 2020. Blender script. Most areas of Blender can be scripted, including animation, rendering, import and export, object creation and automating repetitive tasks (Accessed on 05/16/2020).

Brent Burley, David Adler, Matt Jen-Yuan Chiang, Hank Driskill, Ralf Habel, Patrick Kelly, Peter Kutz, Yining Karl Li, and Daniel Teece. 2018. The design and evolution of disney’s hyperion renderer. ACM Trans. Graph., 37(3).

Angel Chang, Will Monroe, Manolis Savva, Christopher Potts, and Christopher D Manning. 2015. Text to 3D scene generation with rich lexical grounding. arXiv preprint arXiv:1505.06289.

Angel Chang, Manolis Savva, and Christopher D Manning. 2014. Learning spatial knowledge for text to 3D scene generation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2028–2038.

Angel X Chang, Mihail Eric, Manolis Savva, and Christopher D Manning. 2017. Sceneseer: 3D scene design with natural language. arXiv preprint arXiv:1703.00050.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pages 740–750.

Kevin Chen, Christopher B Choy, Manolis Savva, Angel X Chang, Thomas Funkhouser, and Silvio Savarese. 2018. Text2shape: Generating shapes from natural language by learning joint embeddings. arXiv preprint arXiv:1803.08495.

Qi Chen, Qi Wu, Rui Tang, Yuhan Wang, Shuai Wang, and Mingkui Tan. 2020. Intelligent home 3d: Automatic 3D-house design from linguistic descriptions only. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12625–12634.

Yu Cheng, Zhe Gan, Yitong Li, and Jianfeng Gao. 2018. Sequential attention gan for interactive image editing via dialogue.

Bob Coyne and Richard Sproat. 2001. WordsEye: an automatic text-to-scene conversion system. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pages 487–496.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. In Proceedings of EMNLP.

Alaaeldin El-Nouby, Shikhar Sharma, Hannes Schulz, Devon Hjelm, Layla El Asri, Samira Ebrahimi Kakhou, Yoshua Bengio, and Graham Taylor. 2019. Tell, draw, and repeat: Generating and modifying images based on continual linguistic instruction. In The IEEE International Conference on Computer Vision (ICCV).

Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. 2012. Example-based synthesis of 3D object arrangements. ACM Transactions on Graphics (TOG), 31(6):1–11.

Matthew Fisher, Manolis Savva, Yangyan Li, Pat Hanrahan, and Matthias Nießeiner. 2015. Activity-centric scene synthesis for functional 3D scene modeling. ACM Transactions on Graphics (TOG), 34(6):1–13.

Rohit Girdhar and Deva Ramanan. 2020. CATER: A diagnostic dataset for Compositional Actions and TEmporal Reasoning. In ICLR.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems, pages 2672–2680.
Tanmay Gupta, Dustin Schwenk, Ali Farhadi, Derek Hoiem, and Aniruddha Kembhavi. 2018. Imagine this! scripts to compositions to videos. In Proceedings of the European Conference on Computer Vision (ECCV), pages 598–613.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation, 9(8):1735–1780.

Justin Johnson, Agrim Gupta, and Li Fei-Fei. 2018. Image generation from scene graphs. pages 1219–1228.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Isaac V Kerlow. 2009. The art of 3D computer animation and effects. John Wiley & Sons.

Jin-Hwa Kim, Nikita Kitaev, Xinlei Chen, Marcus Rohrbach, Byoung-Tak Zhang, Yuandong Tian, Dhruv Batra, and Devi Parikh. 2017. Co-draw: Collaborative drawing as a testbed for grounded goal-driven communication. arXiv preprint arXiv:1712.05558.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes.

Thomas N Kipf and Max Welling. 2016. Supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Nikhil Krishnaswamy and James Pustejovsky. 2016. Voxsim: A visual platform for modeling motion language. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, pages 54–58.

Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip H. S. Torr. 2019a. Controllable text-to-image generation. In NeurIPS.

Yitong Li, Zhe Gan, Yelong Shen, Jingjing Liu, Yu Cheng, Yuexin Wu, Lawrence Carin, David Carlson, and Jianfeng Gao. 2019b. Storygan: A sequential conditional gan for story visualization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6329–6338.

Yitong Li, Martin Renqiang Min, Dinghan Shen, David Carlson, and Lawrence Carin. 2018. Video generation from text. In Thirty-Second AAAI Conference on Artificial Intelligence.

Rui Ma, Akshay Gadi Patil, Matthew Fisher, Manyi Li, Sören Pirk, Binh-Son Hua, Sai-Kit Yeung, Xin Tong, Leonidas Guibas, and Hao Zhang. 2018. Language-driven synthesis of 3d scenes from scene databases. ACM Transactions on Graphics (TOG), 37(6):1–16.

Elman Mansimov, Emilio Parisotto, Jimmy Ba, and Ruslan Salakhutdinov. 2015. Generating images from captions with attention. CoRR, abs/1511.02793.

Paul Merrell, Eric SchKFuzfa, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. 2011. Interactive furniture layout using interior design guidelines. ACM transactions on graphics (TOG), 30(4):1–10.

Yingwei Pan, Zhaofan Qiu, Ting Yao, Houqiang Li, and Tao Mei. 2017. To create what you tell: Generating videos from captions. In Proceedings of the 25th ACM international conference on Multimedia, pages 1789–1798.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative adversarial text to image synthesis. arXiv preprint arXiv:1605.05396.

Manolis Savva, Angel X Chang, Pat Hanrahan, Matthew Fisher, and Matthias Nießner. 2016. Pirgraphs: learning interaction snapshots from observations. ACM Transactions on Graphics (TOG), 35(4):1–12.

Lee M Seversky and Lijun Yin. 2006. Real-time automatic 3d scene generation from natural language voice and text descriptions. In Proceedings of the 14th ACM international conference on Multimedia, pages 61–64.

Shikhar Sharma, Dendi Suhubdy, Vincent Michalski, Samira Ebrahimii Kahou, and Yoshua Bengio. 2018. Chatpainter: Improving text to image generation using dialogue. ArXiv, abs/1802.08216.

Fuwen Tan, Song Feng, and Vicente Ordonez. 2019. Text2scene: Generating compositional scenes from textual descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6710–6719.

Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X Chang, and Daniel Ritchie. 2019. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. ACM Transactions on Graphics (TOG), 38(4):1–15.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, 13(4):600–612.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funktowicz, et al. 2019. Transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771.

Tao Xu, Pengchuan Zhang, Qiyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. 2018. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. pages 1316–1324.
Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. 2016. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, pages 776–791. Springer.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. 2017. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915.

Song-Hai Zhang, Shao-Kui Zhang, Wei-Yu Xie, Cheng-Yang Luo, and Hong-Bo Fu. 2020. Fast 3d indoor scene synthesis with discrete and exact layout pattern extraction. *arXiv preprint arXiv:2002.00328*.

Xi Zhao, Ruizhen Hu, Paul Guerrero, Niloy Mitra, and Taku Komura. 2016. Relationship templates for creating scene variations. *ACM Transactions on Graphics (TOG)*, 35(6):1–13.

Yang Zhou, Zachary While, and Evangelos Kalogerakis. 2019. Scenegraphnet: Neural message passing for 3d indoor scene augmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7384–7392.