Volume: 9 Issue: VI Month of publication: June 2021
DOI: https://doi.org/10.22214/ijraset.2021.34995

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VI Jun 2021- Available at www.ijraset.com

Texture Oriented Scene Generation from Natural Language Text Description using 3D Coloured Objects

Yashaswini S[1], Shylaja S S [2]
[1]*Cambridge Institute of Technology, Bangalore Urban*
[2]*PES University*

Abstract: To understand language, we need an understanding of the world around us. Language describes the world and provides symbols with which we represent meaning. Still, much knowledge about the world is so obvious that it is rarely explicitly stated. It is uncommon for people to state that chairs are usually on the floor and upright, and that you usually eat a cake from a plate on a table. Knowledge of such common facts provides the context within which people communicate with language. Therefore, to create practical systems that can interact with the world and communicate with people, we need to leverage such knowledge to interpret language in context. Scene generation can be used to achieve an ability to generate 3D scenes on basis of text description. A model capable of learning natural language semantics or interesting pattern to generate abstract idea behind scene composition is interesting [1]. Scene generation from text involves several fields like NLP, artificial intelligence, computer vision and machine learning. This paper focuses on optimally arranging objects in a room with focus on the orientation of the objects with respect to the floor, wall and ceiling of a room along with textures. Our model suggest a novel framework which can be used as a tool to generate scene where anyone without 3D Modeling.
Keywords: 3D Scenes, Semantic model, 3D Modeling.

I. INTRODUCTION

The task of generation of a 3D scene can be interpreted mainly in two ways. First method is the simple drag and drop of individual 3D models according to user requirements. 3D scene design can be very complex using drag and drop method as there are many models to search for and it becomes a cumbersome task to generate correct 3D scene [2].
The second approach would be to visualize and describe the 3D scene using English sentences which are then mapped to our 3D models. The ability to generate a 3D scene with rudimentary English sentences can be simpler as the user just has to describe his visualization and the task of placing and selecting the objects. We consider the second approach where the task of the end user is minimized to a great extent. Text to 3D scene generation has a wide number of applications [3]. It can be used in the educational sector where a complex concept could be described and easily explained through a 3D geometric model.
3D scene generation can be useful even in fields like art or forensics. In this paper, the main focus is on interior designing which has a lot of focus on the orientation of the objects with respect to the floor, wall and ceiling of a room along with textures. Our approach uses Natural Language Processing to
1 is Professor CSE at Cambridge Institute of Technology Bangalore & Research Scholar at VTU and 2 is Head CSE at PES University Bangalore extract useful information from the user text, which will aid the rendering engine to generate a better 3D scene automatically [4]. The above description should make it clear that the 3D mapping of objects in a room is a broad and challenging endeavour. One key factor in the difficulty of this task is that though spatial knowledge is an important aspect of the world it is often not expressed explicitly in natural language. This is one of the biggest challenges in grounding language and enabling natural communication between people and intelligent systems.

II. TASK DEFINITION

We define text-to-scene generation as the task of taking text that describes a scene as input, and generating a plausible 3D scene described by that text as output. More precisely, given an utterance u as input, the output is a scene s: an arrangement of 3D models representing objects at specified positions and orientations in space. To generate scene s, we select objects from a dataset of 3D models and arrange them in likely configurations based on the constraints imposed by the input text u.
There are several challenges in converting a natural language description to a 3D scene. • What constraints are implied by the text?
1) What is the prior knowledge required to represent a scene?
2) How should we ground the objects and spatial relationships?

To address the above challenges, we take natural language and extract physical and spatial constraints expressed which we will refer to as *scene template parsing*. We then take the explicitly specified constraints and expand them through *scene inference*. Finally, we transform the scene template into a physically realizable 3D scene through the process of *scene generation*.
For this to be possible, the system must be able to automatically specify the objects present and their position and orientation with respect to each other as constraints in 3D space. To do so, we need to have a representation of scenes. We need good priors over the arrangements of objects in scenes and we need to be able to ground textual relations into spatial constraints and object presence constraints.

III. PROBLEM STATEMENT

We define the text to scene generation problem as follows: given an utterance u as input, the output is a scene s where s is an arrangement of 3D models representing objects at specified positions and orientations in space. More concretely s = (m_i, T_i) is a set of pairs of 3D model m_i and 3D transformation matrix T_i placing each model within a scene coordinate frame to visually represent the scene. To generate scene s, objects are selected from a database of 3D models M and arranged in likely configurations based on the constraints imposed by the input text u. To represent the constraints expressed by utterance u, I use a graph based semantic representation that I will call a *scene template*. A scene template represents the objects to be visualized as nodes, and constraints between the objects as edges. The scene template is a template from which many concrete geometric scenes can be instantiated.
In this thesis, I will focus on a fairly literal domain of utterances describing common everyday room interiors. For example, the utterance “there is a computer in a living room” is within this scope. We do not handle style or more abstract concepts which require a better understanding of pragmatics (e.g., the utterance “I want a messy child’s bedroom”). Though the presented model is a first step towards inferring unstated, implicit facts, it does not perform iterative Bayesian pragmatic reasoning about the communicative goals of the speaker.
One of the main goals of the current R&D is to be able to handle underspecified natural language input. For instance, in Figure 3.1, the system is able to generate a full scene with inferred kitchen table and chair for the short sentence “there is a sandwich on a plate”. To accomplish this goal, I break down the problem.

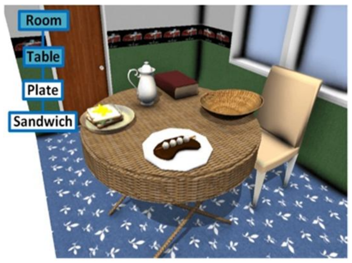
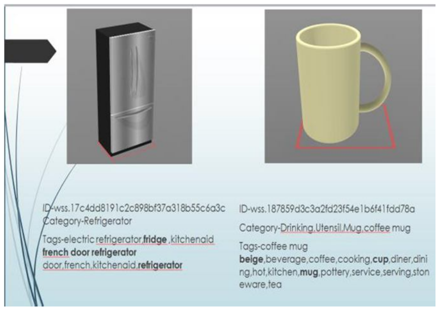


Figure 3.1: Generated scene for “There is a sandwich on a plate.” Note that the room, table and other objects are not explicitly stated, but are inferred by the system.

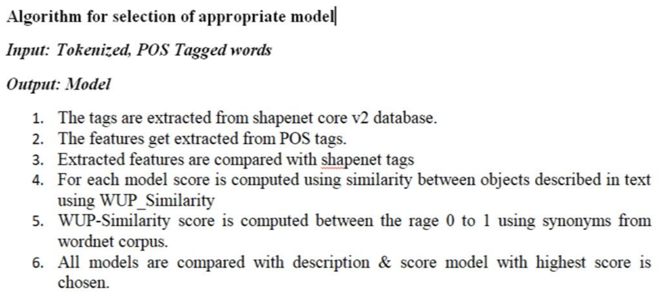
IV. WORK DONE

Implementation was done by analysis of ShapeNetCoreV1 dataset that had few of models which had object file (.obj), material file(.mtl),image files(.jpg and .png) and texture files. The dataset however did not show up texture details properly and even metadata were merely on basic categories (e.g.: when table was queried it showed around 100 model) and tags were empty, hence after a thorough understanding an improvised dataset ShapeNetCoreV2 [6] and ShapeNetSem was considered to extract details like texture and color.
The texture issues which prevailed in v1 were fixed in ShapeNetCoreV2 but still it lacked broader classification of models. Whereas ShapeNetSem had limited models but tags were associated with each and every model, so we started with ShapeNetSem and manually classified data by altering our approach to find broader classes of objects and finally settled with tags as shown in the figure 4.1 each model has ID, category and tags(synonyms) associated with it[7].
Text to scene generation involves processing an input sentence, by identifying items/objects using POS Tagger. The Stanford POS Tagger was used but results were undesired due to wrong classification for some interior designing related words, hence we made our own dataset of nouns and adjectives by identifying features of object related only to interior designing words (nouns).



ID-wss.17c4dd8191c2c898bf37a318b55c6a3c
Category-Refrigerator
Tags-electric refrigerator,**fridge** ,kitchenaid **french door refrigerator** door,french,kitchenaid,**refrigerator**

ID-wss.187859d3c3a2fd23f54e1b6f41fdd78a
Category-Drinking,Utensil,Mug,coffee mug
Tags-coffee mug **beige**,beverage,coffee,cooking,**cup**,diner,dining,hot,kitchen,**mug**,pottery,service,serving,stoneware,tea



Algorithm for selection of appropriate model

Input: Tokenized, POS Tagged words

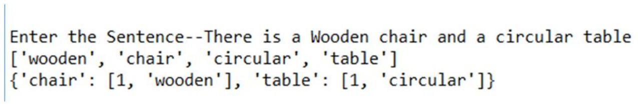
Output: Model

1. The tags are extracted from shapenet core v2 database.
2. The features get extracted from POS tags.
3. Extracted features are compared with shapenet tags
4. For each model score is computed using similarity between objects described in text using WUP_Similarity
5. WUP-Similarity score is computed between the rage 0 to 1 using synonyms from wordnet corpus.
6. All models are compared with description & score model with highest score is chosen.

Fig 4.1 Models associated with ID, Category, tags and algorithm for selection of appropriate model

These features were extracted by converting these words into vectors and compared with tags in database. Model in database are assigned with a score determined by similarity between object, text description and ID. WUP-similarity was used to get similarity score between two objects using WorldNet's synset. These scores can ranges between the values 0-1. All models in datasets are compared with descriptions and scores and model with highest score is chosen and further it is placed in the actual scene. The text entered is processed tokenized, these tokens are searched in dataset of nouns and features are extracted to determine adjective [8].

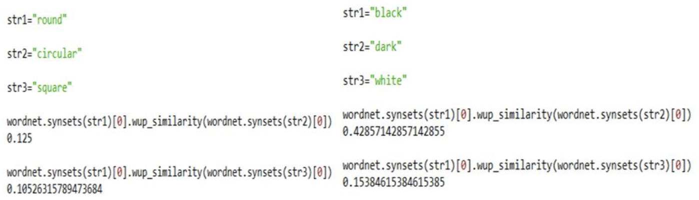
Meanwhile the path of model is stored, its ID, category and tags are stored in an intermediate text file. Finally, a dictionary was created with nouns as key and list of score and adjective as value as shown in the Figure 4.2, for the sentence "there is a wooden chair and a circular table" nouns chair and table are identified along with adjective wooden and circular.
A dictionary created with chair and table as key and score with adjective as values.



```
Enter the Sentence--There is a Wooden chair and a circular table
['wooden', 'chair', 'circular', 'table']
{'chair': [1, 'wooden'], 'table': [1, 'circular']}
```

Fig 4.2 Dictionary created for the input sentence considering noun, adjective and score

The aforesaid dictionary so created was used for the purpose of input sentence considering noun, adjective and score. It is to be borne in mind that the objects require size check before placing it in scene, thus the models are either scaled down /up to their appropriate sizes, rotated and checked as well with other objects to check their relative position. The synonyms are handled using synset and the best probable model is retrieved using WUP similarity as shown in Fig 4.3 for words 'round' and 'black' respectively.



```
str1="round"

str2="circular"

str3="square"

wordnet.synsets(str1)[0].wup_similarity(wordnet.synsets(str2)[0])
0.125

wordnet.synsets(str1)[0].wup_similarity(wordnet.synsets(str3)[0])
0.10526315789473684
```

```
str1="black"

str2="dark"

str3="white"

wordnet.synsets(str1)[0].wup_similarity(wordnet.synsets(str2)[0])
0.42857142857142855

wordnet.synsets(str1)[0].wup_similarity(wordnet.synsets(str3)[0])
0.15384615384615385
```

Fig 4.3 Synset for synonyms for words "round" and "black" with WUP similarity score to retrieve best probable model.

We used Blender as our 3D modeling software since python can be integrated with it and also gives lot of flexibility and freedom compared to other software's like shapenet viewer and mitstuba renderer. Thorough literature survey revealed a model which considers all objects by dividing them into pairs and placing them in order to achieve final scene as shown in Fig 4.4.

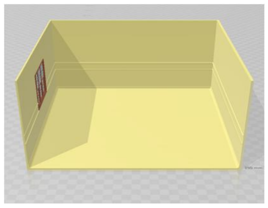
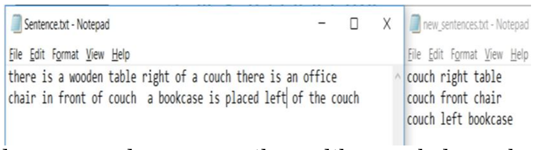


Fig 4.4 3D model of room automatically generated with floor and walls using blender software.

The challenges like scaling factor, relative position and occlusion were addressed by the following approaches : The origin of every object is center of mass, dimension of objects, and these are available in .mtl file which were fetched but the default model were not scaled properly so UNIT attribute was considered for scaling[9]. The placement rules like top, above, below, front, back could not be done using coordinate method as center of mass may not coincide with geometrical origin of objects resulting in condition where objects placed on top may sometime float in air or misplacement of objects as shown in Fig 4.5.

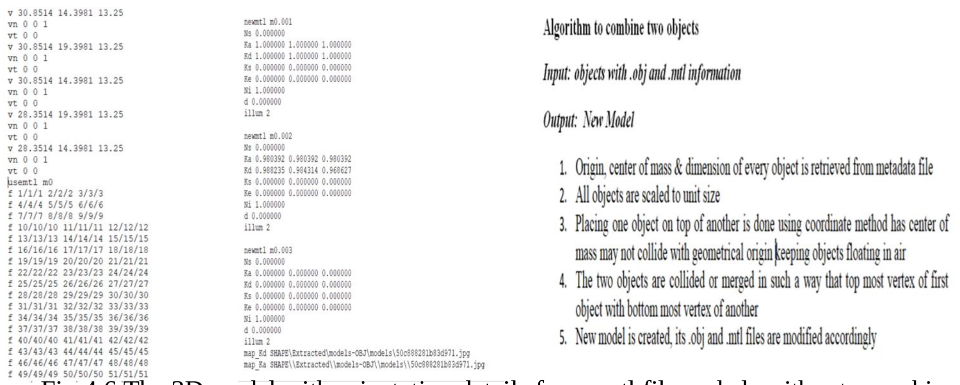

Sentence.txt - Notepad
File Edit Format View Help
new_sentences.txt - Notepad
File Edit Format View Help

```
there is a wooden table right of a couch there is an office
chair in front of couch  a bookcase is placed left of the couch
```

```
couch right table
couch front chair
couch left bookcase
```

Fig 4.5 Simplifying the placement rules w.r.t attributes like top, below, above right and left for object placement.

To solve this problem we took two objects and collided/merged top most vertex of top object with bottom most vertex of bottom object. This resulted in creation of new models by manipulating .obj and .mtl files as shown in Fig 4.6.



```
v 30.8514 14.3981 13.25
vn 0 0 1
vt 0 0
v 30.8514 19.3981 13.25
vn 0 0 1
vt 0 0
v 30.8514 14.3981 13.25
vn 0 0 1
vt 0 0
v 28.3514 19.3981 13.25
vn 0 0 1
vt 0 0
v 28.3514 14.3981 13.25
vn 0 0 1
vt 0 0
usemtl m0
f 1/1/1 2/2/2 3/3/3
f 4/4/4 5/5/5 6/6/6
f 7/7/7 8/8/8 9/9/9
f 10/10/10 11/11/11 12/12/12
f 13/13/13 14/14/14 15/15/15
f 16/16/16 17/17/17 18/18/18
f 19/19/19 20/20/20 21/21/21
f 22/22/22 23/23/23 24/24/24
f 25/25/25 26/26/26 27/27/27
f 28/28/28 29/29/29 30/30/30
f 31/31/31 32/32/32 33/33/33
f 34/34/34 35/35/35 36/36/36
f 37/37/37 38/38/38 39/39/39
f 40/40/40 41/41/41 42/42/42
f 43/43/43 44/44/44 45/45/45
f 46/46/46 47/47/47 48/48/48
f 49/49/49 50/50/50 51/51/51
```

```
newmtl m0.001
Ns 0.000000
Ka 1.000000 1.000000 1.000000
Kd 1.000000 1.000000 1.000000
Ks 0.000000 0.000000 0.000000
Ke 0.000000 0.000000 0.000000
Ni 1.000000
d 0.000000
illum 2

newmtl m0.002
Ns 0.000000
Ka 0.980392 0.980392 0.980392
Kd 0.988235 0.984314 0.968627
Ks 0.000000 0.000000 0.000000
Ke 0.000000 0.000000 0.000000
Ni 1.000000
d 0.000000
illum 2

newmtl m0.003
Ns 0.000000
Ka 0.000000 0.000000 0.000000
Kd 0.000000 0.000000 0.000000
Ks 0.000000 0.000000 0.000000
Ke 0.000000 0.000000 0.000000
Ni 1.000000
d 0.000000
illum 2
map_Kd SHAPE\Extracted\models-OBJ\models\50c888281b83d971.jpg
map_Ka SHAPE\\Extracted\\models-OBJ\\models\\50c888281b83d971.jpg
```

Algorithm to combine two objects

Input: objects with .obj and .mtl information

Output: New Model

1. Origin, center of mass & dimension of every object is retrieved from metadata file
2. All objects are scaled to unit size
3. Placing one object on top of another is done using coordinate method has center of mass may not collide with geometrical origin keeping objects floating in air
4. The two objects are collided or merged in such a way that top most vertex of first object with bottom most vertex of another
5. New model is created, its .obj and .mtl files are modified accordingly

Fig 4.6 The 3D model with orientation details from .mtl file and algorithm to combine two objects

Further, the colour of the model can be changed by changing the values of RGB values between the range 0 to 255 and the dimension of new model generated is fixed as 800 X 800 using image processing package PIL as explained in the Figure 4.7.

Algorithm to change object color

Input: Model

Output: New Model with color changed

1. The .mtl file consists of basic color and texture of model
2. Using image processing PIL package the models are read and default RGB values are noted
3. RGB values of model are altered according to desired color
4. New model has color information updated in both geometrical and texture file in dataset.

Fig 4.7 Applying Color to the 3D object with Algorithm.

In the current work, the texture information is extracted from the .mtl file and the rooms interiors are modified as shown in the Figure 4.8. There are different textures applied to wall, floor and the wooden items which converts the room layout to more realistic structure[10]. Even though the texture is applied to the room surface, when attempt was made to apply texture to objects it failed due to lack of processing power to generate scene.

Fig 4.8 Texture information extracted from .mtl file

V. PITFALLS

The drawback of scene generation is input text i.e. sentences should be straight forward or more descriptive to achieve good results. produce desired results. Due to scaling complexities the objects may exceed dimensions of room. Multiple objects in same scene can create relative reference and occlusion effects on generated scenes as shown in the Figure 4.9. There needs to be more options given to users in terms of texture, color of objects, so that a coloring and texture applying model could be implemented to create realistic scenes of interior designing[11].

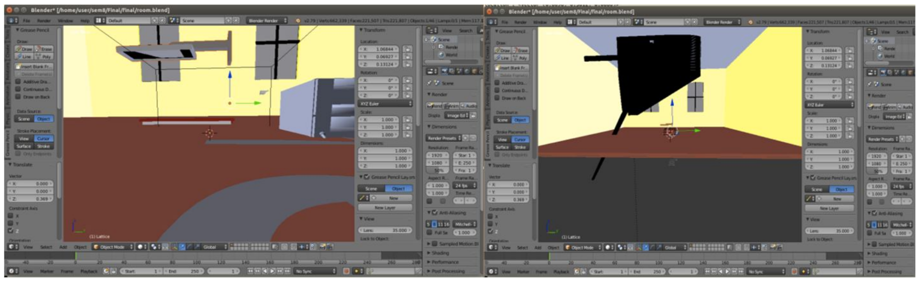


Fig 5.9 : Blender output for the sentence "book on shelf"

The output shown in Fig 5.9 shows generates the inappropriate scene instead of displaying "the book kept on shelf "the bookshelf itself is displayed in scene [12]. The implicit understanding is lacking, hence the quality of scene and semantic of sentences can be still improved. The pattern can be applied to the wall floor of the room and object having the texture details can also be placed in the scene, however all the objects do not have texture details in material file (.mtl file) in ShapenetCorev2 dataset.

VI. RESULTS

The results obtained from the current R&D is elucidated as under:

A. Scene 1

The scene generated for the sentence "There is a white computer chair to the right of couch. There is a wooden bed to left of couch, table is in front of wooden bed, a laptop is kept on the table". In the scene we can note the textures are applied to wall floor respectively and the scene generated is more realistic as shown in Fig 6.1, we can note that the bed placed is outside the room, hence it requires boundary checking.

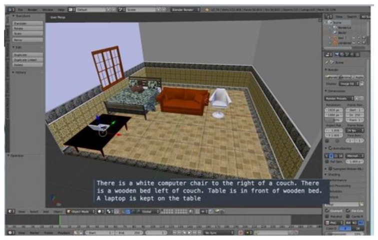

There is a white computer chair to the right of a couch. There is a wooden bed left of couch. Table is in front of wooden bed. A laptop is kept on the table

Fig 6.1 Scene generated by placing objects in 3D room by calculating relative position

B. Scene 2

The scene generated for the sentence "there is a wardrobe left of white leather couch, there is a glass table in front of couch, a fruit is kept on table, there is white computer chair to left of couch". In the scene objects seems to be aligned w.r.t the assumed coordinates here wooden table and white bed is displayed, but the chair is picked randomly without applying the texture details as shown in Fig 6.2

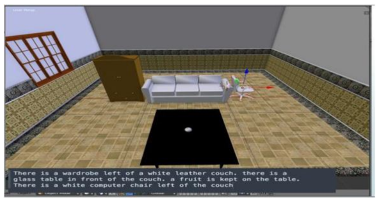

There is a wardrobe left of a white leather couch. there is a glass table in front of the couch. a fruit is kept on the table. There is a white computer chair left of the couch

Fig 6.2 Scene generated by objects bound checking w.r.t. room dimensions

C. Scene 3

The scene generated for the sentence "there is a wooden table in front of white bed, there is an office chair to right of table".
In the scene objects seems to be aligned w.r.t the assumed coordinates here wooden table and white bed is displayed, but the chair is picked randomly without applying the texture details as shown in Fig 6.3.

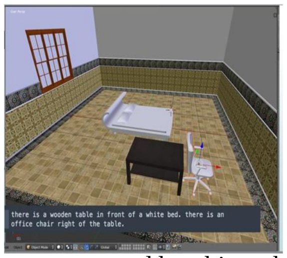

there is a wooden table in front of a white bed. there is an office chair right of the table.

Fig 6.3 Scene generated by objects by explicit reference

D. Scene 4

The scene generated for the sentence "there is a glass table to right of wooden bed, there is leather couch to left of wooden bed, a lamp on table and white chair is in front of glass table"
In the scene objects seems to be aligned w.r.t the assumed coordinates here the leather couch is placed outside the boundary of room as shown in Fig 6.4. The objects fail to calculate boundary of room.

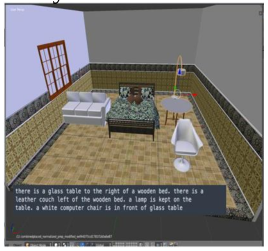

there is a glass table to the right of a wooden bed. there is a leather couch left of the wooden bed. a lamp is kept on the table. a white computer chair is in front of glass table

Fig 6.4 Scene generated for multiple objects

VII. EMPIRICAL EVALUATION

We evaluate the output of our system by asking people to judge how well generated scenes match given input descriptions. This is an appropriate initial evaluation since in a practical usage scenario, a scene matching the input description well would provide a good starting point for further refinement. We compare versions of our system contrasting the benefit of different components. We also establish an upper-bound baseline by asking people to manually design scenes corresponding to the same descriptions, using a simple scene design interface used in prior work (Fisher et al., 2012).

VIII. CONCLUSION

This work mainly focuses on getting appropriate 3D models when user inputs a seed sentence. In a natural language, not all constraints can be explicitly specified. In this work we generate new 3D models by collapsing the geometric coordinates of two objects to solve the parent child relationship issues. The texture and color information are extracted from .mtl file, the color can be changed by using RGB coordinates ranging the value from 0 to 255.
In order to address the challenge of selecting appropriate object the synonyms are extracted from sysnsetID. The adjective occurring before noun is extracted to apply texture to shapenet models. However, the objects are placed in random dimensions has the spatial information are not handled by this work.

IX. WAY AHEAD

The Authors et al, opine that the following features can be further added on to the current work: Default texture can be applied for object without .mtl information.

A. Considering the object size w.r.t to room dimension
B. The number of objects placed at a time in single frame should be decided to avoid occlusion.

REFERENCES

[1] Angel Chang, Will Monroe, Manolis Savva, Christopher Potts, and Christopher D. Manning. "Text to 3D Scene Generation with Rich Lexical Grounding". ACL 2015.
[2] Angel X. Chang, Manolis Savva, and Christopher D. Manning. Learning Spatial Knowledge for Text to 3D Scene Generation. EMNLP 2014.
[3] Angel X. Chang, Manolis Savva, and Christopher D. Manning. Interactive Learning of Spatial Knowledge for Text to 3D Scene Generation In Proceedings of the ACL 2014 Workshop on Interactive Language Learning, Visualization, and Interfaces (ACL-ILLVI).
[4] Angel X. Chang, Manolis Savva and Christopher D. Manning. Semantic Parsing for Text to 3D Scene Generation.
[5] Bob Coyne and Richard Sproat. WordsEye: An Automatic Text-to-Scene Conversion System. 2001.
[6] Kristina Toutanova, Dan Klein, Christopher D. Manning and Yoram Singer 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network
[7] Kristina Toutanova and Christopher D. Manning 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger.
[8] Peilu Wang, Yao Qian, Frank K. Soong, Lei He, Hai Zhao1, Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network for conference at arXiv on oct 2015
[9] Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit in Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.
[10] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 363-370.
[11] Danqi Chen and Christopher D Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. Proceedings of EMNLP 2014.
[12] Richard Socher, John Bauer, Christopher D. Manning and Andrew Y. Ng. 2013. Parsing with Compositional Vector Grammars. Proceedings of ACL 2013.