Kalpayita - A Machine Learning Approach to Interior Designing

Komma Karthik Reddy[1] Apoorva Jaiswal[2] Ashish Agarwal[3] Akshaya S.[4] Dr. Krishnan R[5]
[5]Professor
[1,2,3,4,5]Department of Computer Science & Engineering
[1,2,3,4,5]Dayananda Sagar College of Engg., Bengaluru, India

Abstract— The existing approaches in Interior Designing do not leverage the latest technologies such as Machine Learning and are stuck with mundane methods like drag and drop approach to create a required scene. SceneSeer is one of the few existing systems which makes use of Text to 3D scene conversion approach, but it is still not used much in the field of Interior Designing. We present Kalpayita: A Voice driven tool to make Interior Designing hassle free. It converts voice to 3D scene. Voice commands in natural language allow the users, such as Interior Designers or even the people with least knowledge in this field, to interact with the system. These commands are converted into text and Natural Language Processing is applied on the text to make the system understand what the user wants. The result generated from Natural Language Processing is then fed into the jMonkeyEngine to display the 3D scene. Our system is an improvement over SceneSeer. We make use of voice commands, and our studies show that the accuracy and time complexity of our system Kalpayita is much better than the pre-existing Text to 3D scene conversion systems. Visual impact is an important aspect in Interior Designing. Likewise, our system generates 3D scenes which are better than other systems in aesthetics and looks.

Key words: jMonkey, Natural Language Processing, Interior Designing, Machine Learning

I. INTRODUCTION

State of the art Interior designing systems use obsolete technologies like drag-n-drop [4]. Few other advanced systems like WordsEye [3] involve textual commands to generate a scene but use language that describes explicit and implicit constraints which is unusual and difficult to use for humans. SceneSeer [1] is a system that uses Natural Language text to generate a 3D scene. This lets the users to work with the system in easier way. Our system Kalpayita is an advancement and implementation of the SceneSeer . We provide a voice interface and improve on the time efficiency of the generated scene. The accuracy with which attributes of objects and relations between two different objects are captured, is also improved.

In this system, the input voice command is given in natural language which is then converted to text using Google Speech API. We use the NLP based text understanding to understand the scene descriptions and then apply probability based approaches to further understand the relation between the objects [2]. This is then used to collect the objects and their texture, size and colour from the dataset to create a 3D scene [6] from it which is rendered by jMonkey Engine.

The remainder of the paper is organized as follows. Section 2 tells about the relevant literature study. In Section 3, the system architecture and implementation is explained. In Section 4, the conducted experiments are discussed, including the results in Section 5. In Section 6, the discussion and conclusion is presented.

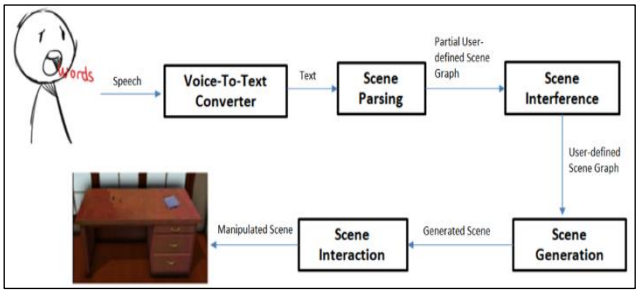

Fig. 1: System Architecture

II. RELEVANT LITERATURE

The previous interior designing approaches [4] uses drag and drop techniques with adjustable properties. WordsEye [3] relies on a large database of 3D models and poses to depict entities and actions. Detailed instructions need to be given, so no implicit understanding is possible. Natural Language Processing has been used to understand the text given as input by any common user. Spatial Knowledge [2] is also used in earlier approaches for scene generation, where grounding objects are found and objects are placed to satisfy all constraints. Determining the support hierarchy, supporting surfaces and spatial relations is also used for scene generation [1]. Other work does scene synthesis [6] by training occurrence and arrangement models. Semantic Parsing [5] is used where textual input is parsed provided by the user into a sequence of commands with relevant parts of the scene as arguments, relying on hand-coded rules for text to 3D scene generation.

III. SYSTEM ARCHITECTURE AND IMPLEMENTATION

A. System Overview

The system is composed of five modules, each of which leverages diverse tasks for creating a 3D scene out of the spoken input scene description. The Fig. 1 shows overview of the system's architecture.

B. Voice-to-Text conversion

This module makes use of the Google Speech API for speech to text conversion. The API needs voice input in FLAC format. The voice is recorded in WAV format and then it is converted into FLAC format for usage. The FLAC format makes sure that the voice quality is not degraded and thus accuracy of text conversion is maintained.

C. Scene Parsing

This is the first module in the text processing. The objects, attributes and spatial relations are identified from the input text. The text is processed using the Stanford CoreNLP pipeline [1]. To identify objects, nouns are extracted first using Stanford CoreNLP pipeline and then sent to the WordNet system to identify the visible nouns, which are considered as physical objects.

The other nouns and the adjectives are considered to be attributes for the objects [5]. In some cases, visible nouns are also captured as attributes by the Stanford CoreNLP pipeline. Also, an attribute can refer to more than one object in a sentence. For example, for a sentence, "The chair and desk have wooden material", attribute wooden applies to both chair and desk. For such cases, the use of a simple algorithm does the job. The algorithm is designed keeping in mind the pattern with which the words are spoken. If two nouns occur consecutively, the first noun here is considered as an attribute. For a single sentence, the latest attribute (adjective) in that sentence is applied to all following objects in that sentence, if the objects are not being provided with any other attribute. After finding the attribute, the attribute type is computed using WordNet, one among the four types: material, color, pattern and shape.

For finding spatial relations, we see that another simple algorithm gives much better results compared to the method being used by SceneSeer [1], which used Semgrex pattern to do their task. In our system, the spatial relation between two objects is computed by observation of specific words and their position of occurrence with respect to the two objects. The spatial relations include left_of, right_of, top, bottom, and so on. After this step, we have a partial scene graph depicting the objects-attribute pairs and spatial relations between objects.

D. Scene Inference

This step involves finding out missing objects in the input text. The input generally remains in natural language and thus the implicitly mentioned objects need identification. Naïve Bayes' Machine Learning approach is used for this purpose. It uses 1024 seed sentences for learning. The new input text is then queried to identify the scene type. We make use of a set of child and parent objects for each scene type possible to identify the possible implicit objects. For each object, the support object is identified. All these tasks use separate Naïve Bayes models. These objects are included in the set of objects, if not already present. The following formulas [2] are used for the Naïve Bayes models:

1) *Probability of parent object C_p given child object C_c:*

$$P(C_p|C_c) = \text{count}(C_c \text{ on } C_p) / \text{count}(C_c)$$

2) *Probability of Scene type S_n given the child object C_c:*

$$P(S_n|C_c) = \text{count}(C_c \text{ on surface with } S_n) / \text{count}(C_c)$$

3) *Probability of the occurrence of a given object C_o given the scene type C_s:*

$$P_{occ}(C_o|C_s) = \text{count}(C_o \text{ in } C_s) / \text{count}(C_s)$$

A scene graph is created based on the hierarchy of objects and the spatial relations between objects. The graph's root is the room object, followed by the children as the objects that are directly placed on the room. The objects in the same level have edges labeled as the spatial relation between the objects. All the objects, in the text and ones that were inferred, are arranged in a scene graph, such that the supported objects are taken in a different level in the scene graph. It is depicted in Fig. 2.

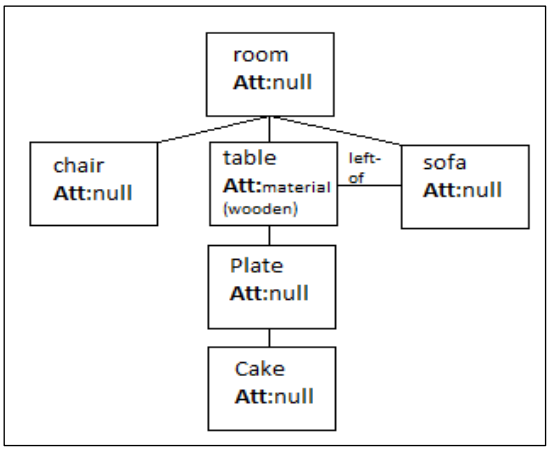

Fig. 2: User defined Scene Graph- "There is a room with a wooden table and a sofa. The table is to the left of the sofa. There is a cake on the table."

E. Scene Generation

This module involves rendering the objects identified in the previous steps with their attributes and spatial relations to form the complete 3D scene [6].

The jMonkey Engine is used for rendering the 3D scene in the system. The scene graph created in the Scene Inference step is traversed in depth-first order to place support objects relative to their support. The spatial relations are then resolved by relatively positioning related objects. Finally, the independent objects in each level of the scene graph is traversed and placed randomly, ensuring that the objects are not overhanging or colliding.

While placing objects, the best object is picked from the database based on the attribute and the object type, and suitable attributes are applied. Since randomizing placements may not always give good results, user is given an option to re-randomize the placements any number of times. Fig 3 shows how the scene graph is generated in jMonkey during implementation. Node and Spatial are types defined in jMonkey, represented here by rectangle and oval respectively.

F. Scene Interaction

This module helps the user in interacting with the system. The generated scene can be modified in this step.

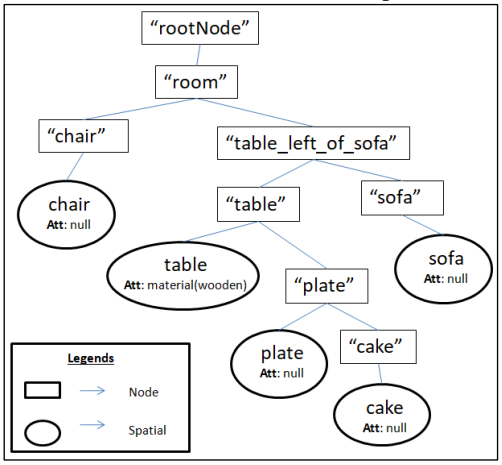

Fig. 3: Complete Scene Graph after Scene Generation - "There is a room with a wooden table and a sofa. The table is to the left of the sofa. There is a cake on the table."

Voice is converted to text and then processed to identify the type of modification. The objects and attributes along with newly identified spatial relations specified in the input are identified. The generated scene is taken into consideration to make sure minimum changes are made to it.

The insert command involves altering the scene graph to include new object with its attribute and any spatial relation with any existing objects. The scene graph is traversed again and then the parent object for the current object is identified using the Naïve Bayes model that was used in the inference stage. The parent Node is determined and the new child Node is attached to it. The new Scene graph is then rendered.

The move command provides a new translation to the object mentioned. This involves scene graph traversal and replacement of existing translations with new ones.

The delete command involves traversing the scene graph, and deleting the object Node from the scene graph.

The enlarge and shrink commands involve scene graph traversal and scaling the found object. Enlarge command uses a scaling factor greater than 1, and shrink command uses a scaling factor less than 1.

IV. EXPERIMENTS

To evaluate the usefulness of our approach and the accuracy of implementation, we test the system with all possible commands. The general command to generate the scene is given first followed by the possible randomization or modification commands.

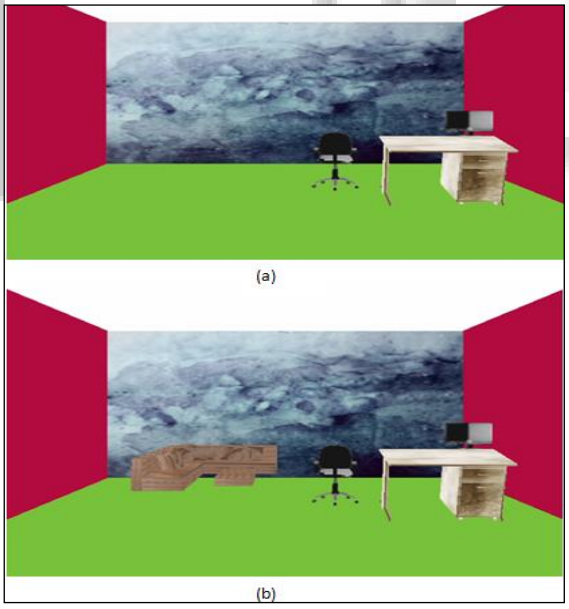

Fig. 4(a): Generate- "There is a monitor" (b) :Modification- "Insert a wooden sofa."

Few generated scenes in the process are shown in the Fig. 4. The users could use the following commands during the experiment:

1) Generate - A scene is generated after speech is converted into text.
2) Move <position>- The specified object is moved to the given position during scene interaction phase. e.g.: move right, move back, etc.
3) Insert - The specified object is inserted during scene interaction phase.
4) Shrink - The size of the specified object is reduced if not satisfied during scene interaction phase.
5) Enlarge - The size of the specified object is enlarged during scene interaction phase.
6) Randomize- A random scene is generated after the speech to text conversion.
7) Delete - The specified object will be deleted if not needed during the scene interaction phase.

V. RESULTS

The experiment was repeated with 154 people and they were told to generate 3 scenes each and then rate these scenes on a scale of 10. Each rating was noted and we got an average rating of 7.2 per scene. The standard deviation in our results was calculated to be 1.4. The Fig 5 and 6 show two such scenes and the ratings given by the user.

The tabulation of results has been shown in the table 1.

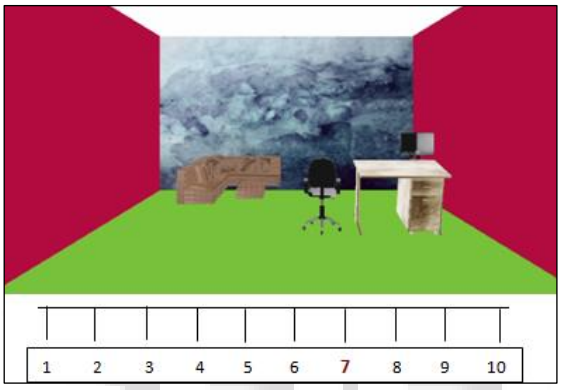

Fig. 5: User rating for the modified scene in the fig 4.

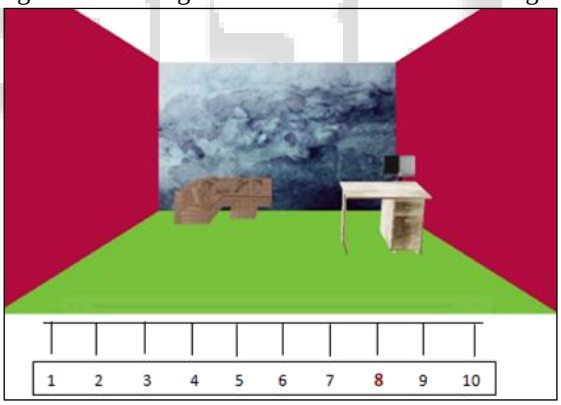

Fig. 6: User rating for the scene in fig 4 modified by the command, "Delete the chair"

VI. CONCLUSION

The tool Kalpayita, built keeping SceneSeer as the base, has been through some improvements fetching us better results. Kalpayita is time efficient and easy to use. The voice interface provides for a better user experience and easier modification. The randomization option gives it a more realistic look and feel.

The future advancements for the system include providing the interior designer with interface to add more objects based on their choice, let the user drag and change existing positions of objects, adding lighting and shadow effects for better feel of interior designing, recording the modifications done by the user to improve the randomized

scene generation and recording user feedbacks on each generated scene for better generation.

No. of users rating the generating and rating the scenes	154
No. of scenes shown per user	3
Average rating per scene	7.2
Standard Deviation	1.4

Table 1: Result tabulation for the experiment

REFERENCES

[1] A. X. Chang, M. Eric, M. Savva and C. D. Manning, "SceneSeer: 3D Scene design with natural language", ACL(2017)
[2] A. X. Chang, M. Savva, and C. D. Manning, "Learning spatial knowledge for text to 3D scene generation", in Proceedings of Empirical Methods in Natural Language Processing (EMNLP)
[3] B. Coyne, R. Sproat, "WordsEye: An automatic Text-to-Scene Conversion System", in SIGGRAPH(2001)
[4] S. H. Saw, H. W. Tan, S. Safdar, T. B. Tan, "A Simple Interactive 3D Interior Design Application For Living Room", IEEE(2015)
[5] A. X. Chang, M. Savva and C. D. Manning, "Semantic Parsing for Text to 3D Scene Generation", ACL (2014)
[6] M. Fisher, D. Ritchie, M. Savva, T. Funkhouser, P. Hanrahan, "Example-based Synthesis of 3D Object Arrangements", ACM (2012)

