A Constraint-based Text-to-Scene Conversion System

Liuzhou Wu
School of Computer Science and Engineering
South China University of Technology
Guangzhou, China
Email: liuzhouwu@163.com

Zelin Chen
School of Computer Science and Engineering
South China University of Technology
Guangzhou, China
Email: cszlchen@scut.edu.cn

Abstract—This paper describes a system called Text2Scene for automatically converting text into 3D scenes. Text2Scene relies on a large database of 3D models to depict entities and actions. Every 3D module can have associated information, such as spatial tags, color and functional properties for depiction process. Text2Scene takes the implicit and conflicting constraints into account to generate reasonable scenes.

Keywords-text-to-scene; natural language processing; depiction; constraints; scene generation

I. INTRODUCTION

Nowadays 3D graphics are used in many applications, such as cartoons, animations and games. However, creating 3D graphics is a difficult and time-consuming task. The user must learn to use a complex software package before he/she can actually create the artwork. A new paradigm which makes the creation of 3D graphics effortless and convenient is needed. It should be possible to describe 3D scenes directly from natural language.

The conversion of natural language text into graphics has been investigated in a few projects. NALIG[1] is an early example of them that was aimed at recreating 2D scenes. One of its major goals was to study the relationship between space and prepositions. NALIG considered simple phrases in Italian of the type subject, preposition, and object that in spite of their simplicity can have ambiguous interpretations. The Put system [2] which shared our goal of making graphics creation easier was limit to spatial arrangements of existing objects. And what’s more, its input was restricted to the form Put (X, P, Y), where X and Y were objects, and P was a spatial preposition. The Carsim system [3][4] which converted a traffic accident report to 3D scene can not generate scenes outside the traffic field.

Our Text2Scene, in contrast, is to provide a blank slate where the user can literally paint a picture with words. The description may consist not only of spatial relations, but also actions performed by objects in the scene.

II. OVERVIEW OF SYSTEM

Fig. 1 shows the architecture of Text2Scene.

When an input text is entered, the sentences are tagged and parsed, the output of the parser is then converted to a dependency structure, and this dependency structure is then semantically interpreted and converted into a semantic representation.

Depiction rules are used to convert the semantic representation to a set of low-level depictors representing 3D objects, poses, spatial relations, color attributes, etc; note that a pose can be loosely defined as a character in a configuration suggestive of a particular action. Transduction rules are applied to resolve conflict and implicit constraints. The resulting depictors are then used to manipulate the 3D objects that constitute the final, renderable 3D scene. Fig. 2 shows an example of converting a short text to graphical scene.
III. NATURAL LANGUAGE PROCESSING

Natural language processing module works as the following two steps.

A. Dependency Representation

The input text is initially tagged and parsed using a part-of-tagger [5] and a statistical parser [6]. The output is a parse tree that represents the structure of the sentence. The parse tree is then converted into a dependency representation which is simply a list of the words in the sentence, showing the words that they are dependent on (the heads) and the words that are dependent on them (the dependents). Fig. 3 shows an example of dependency representation.

B. Semantic Representation

The next phase of the analysis involves converting the dependency structure into a semantic representation. The semantic representation is a description of the entities to be depicted in the scene, and the relationship between them [7]. The semantic representation for the sentence “Mark said that the cat was under the chair” is like this:

```
("node2" :ENTITY :3D-OBJECTS ("boy")
 :LEXICAL-SOURCE "Mark" :SOURCE SELF)
("node1" :ACTION "say" :SUBJECT "node2"
 :DIRECT-OBJECT ("node5" "node4" "node7")...)
("node5" :ENTITY :3D-OBJECTS ("cat"))
("node4" :STATIVE-RELATION "under" :FIGURE "node5"
 :GROUND "node7")
("node7" :ENTITY :3D-OBJECTS
 ("table-vp14364" "nightstand-vp21374"
  "table-vp4098" "pool_table-vp8359" ...)...))
```

The semantic representation is a list of semantic representation fragments, each fragment corresponding to a particular node of the dependency structure.

IV. DEPICTION MODULE

In this module, high-level semantic representation, which is the output of natural language processing module, are converted to low-level 3D objects and relations.

A. Depictor

All scenes are ultimately defined in terms of a set of low-level graphical specifications called depictors [8]. We use depictors to control the property of 3D object, such as size, color, position, transparency and so on.

Text2Scene has approximately 1,500 3D polygonal objects at present, with another 10,000 in the process of being integrated into the system. Most of the objects in the database are models for human characters, animal, building, vehicles, household items, plants and so on. Users can add their own models into the database to extend the system.

Text2Scene has not only the raw 3D models, but also the associated information of the models, as listed in Table 1.

Most types of information are easy to understand, while we should explain spatial tags. We depict spatial relation between objects by associating spatial tags with the objects in the database. For example, the interior area of a pan functions as a cup to contain something in it. So the area is marked with a “cup” tag, and is used to depict spatial relations such in and on. We define some other spatial tags, such as top, base, canopy, ridge, peak and so on. Fig. 4 shows spatial tags for “canopy”.

We use the spatial relation between objects to define the basic layout of scenes. There are two types of spatial relation, one is relative position, such as “A chair is next to the table”, the other is distance, such as “The dog is 5 feet away from the refrigerator”.

Spatial relation is often denoted by prepositions like on, under, beyond and so on. For example, “The bird is on the cat”; we find a top surface tag for the cat (on its back) and a base tag for the bird (under its feet). We then reposition the bird so that its feet are on the cat’s back.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default size</td>
<td>All objects are given a default size</td>
</tr>
<tr>
<td>Skeletons</td>
<td>Used in human and animal characters to define poses when representing actions</td>
</tr>
<tr>
<td>Shape</td>
<td>Used in some objects that can change shape to depict states of the objects</td>
</tr>
<tr>
<td>Parts</td>
<td>These are named collections of polygonal faces representing significant areas of the surface. For example, the headlights, roof and windshield of a car</td>
</tr>
<tr>
<td>Color</td>
<td>These are the set of parts to be colored when the object is specified by the text as having a particular color.</td>
</tr>
<tr>
<td>Transparency</td>
<td>These are parts that get a default transparency.</td>
</tr>
<tr>
<td>Spatial tags</td>
<td>Used to depict spatial relations.</td>
</tr>
<tr>
<td>Functional properties</td>
<td>Used in the depiction process to determine how an object can be used.</td>
</tr>
</tbody>
</table>

![Spatial tags for "canopy"](Image)
B. Depiction Processing

We just talk about the low-level depicter (3D objects and their spatial and graphical properties). The main task of depiction module is to translate the high-level semantic representation produced by the linguistic analysis into low-level depicter. The process is outlined as below:

1. Convert the semantic representation produced by the linguistic analysis to a list of typed semantic elements with all references resolved. Find out “what?”, “when?”, “where?”, and “how?” from the semantic representation.
2. Read in referenced 3D models from database and assign depictors to each semantic element, with associated spatial information and other information of the elements.
3. Resolve implicit and conflicting constraints of depictors.
4. Apply each assigned depictor, while maintaining constraints, to incrementally build up the scene.
5. Add background environment, ground plane, lights, adjust the camera if necessary.
6. Render and generate scene.

C. Implicit and Conflicting Constraints

In certain circumstances it is desirable to add depictors for constraints that are not explicitly stated, but rather are based on common sense knowledge or are in some way deducible from the semantic representation. A set of transduction rules is invoked to do this.

Consider: “The book is next to the laptop”. Although it is not stated that the book is on the table, we probably want it there rather than floating in the air next to the laptop. To do this, we invoke a rule that says “If X is next to Y, X is not already on a surface, and X is not an airborne object (e.g., a helium balloon)” then “Put X on the same surface as Y”.

Depiction specifications sometimes conflict with one another. This typically occurs when the default depictors assigned by an action conflict with those explicitly specified elsewhere. For example, “Tom waved to Lucy” will generate depictors to put Tom in a waving pose, put Tom behind and facing Lucy, etc. Some of those depictors, such as the exact positions of the two characters, are labeled tentative because they are just defaults and are not inherent to the waving pose. So we get the tentative constraint:

Constraint1: (POSITION: Tom 15 feet from Lucy in Z axis [tentative])

But assume we add the specifications that “Lucy is 20 feet to the left of Tom and Lucy is 30 feet behind Tom”, which generates new constraints:

Constraint2: (POSITION: Lucy 20 feet from Tom in X axis)

Constraint3: (POSITION: Lucy 30 feet from Tom in Y axis)

We now have conflict. To resolve this, a transduction rule is invoked that when detecting conflict between depictors X and Y, where X is tentative, depictor X will be removed. In this example, since constraint1 is marked as tentative, it is removed.

D. Applying Depictors

Text2Scene applies various independent depictors (such as poses, spatial relation and so on) to generate a realistic scene. It is done by applying constraints in the following way:

1. Initialize objects with default size and shape.
2. Change the size and shape according to the constraints.
3. Position all the objects in the right place.
4. Determine the orientation for each object.

V. DISCUSSION AND FUTURE WORK

Text2Scene provides a new approach for users to creating 3D scenes quickly. It reads from natural language input, then converts it into scenes. From the results in our experiments, Text2Scene is efficient and robust. One of the key features is that Text2Scene can handle the implicit and conflicting constraints quite well. However, there are many areas where Text2Scene needs to be improved, such as: improvement in the coverage and robustness of the natural language processing; language input via automatic speech recognition rather than text; generate dynamic animation rather than static scenes.

REFERENCES