VRML for automatic generation of 3D Scene

Priyanka Jain*, R. P. Bhavsar**, B. V. Pawar**, Hemant Darbari*

*Center for Development of Advanced Computing, Pune, India  
**North Maharashtra University, Jalgaon, India

ABSTRACT

The broader objective of our research is to work for automatic text visualizer (ATV) for Hindi language named ‘Preksha’, which forms a scene by understanding a text of Hindi language. This research scope for ATV is expending with language processing, knowledge processing, and scene processing to generate the virtual environment. After presenting an overall architecture of an ATV engine in brief, a design is proposed for scene planning using scene grammar. However, the language considered for research is Hindi but the proposed scene grammar is language independent and extendible. It is explained with examples using Virtual reality Modeling Language (VRML). The contribution of this paper is to work toward scene synthesis and adding behavior for the generation of 3D scene.

Key words: Automatic text visualization (ATV), Natural language processing (NLP), Natural Language Visualization (NLV), Automatic Scene generation, Virtual reality (VR).

Corresponding Author: Priyanka Jain

INTRODUCTION

Automatic Text visualization (ATV), i.e. scene generation based on natural language understanding, is a new interdisciplinary research area based on the work in natural language processing, and computer graphics. This process covers many disciplines including linguistic, computational linguistics, psycholinguistic, philosophy, computational graphics, virtual reality, and cognitive science. An important challenge in ATV is to compare the visualization result with human cognition. This is a problem in real life also, where different readers process, comprehend and visualize the same information with their own perspectives and cognition abilities. Similarly, the automatic runtime-generated scene by an ATV system may or may not be exactly mapped with the scene imagined by the user. There are many-to-many relations in the linguistic semantic and its visualization. A single scene generated from a text can have multiple representations. As two translators translate the text in their own way (may it be automatic or manual), similarly, the information visualized by two different visualizers will be different. It will be based on user’s visualizing power and past experience. The focus of this paper is on planning scene grammar and generating 3D scene using extracted knowledge from Hindi language processing. Although, the system is targeted for the
In this Section-1 named ‘Introduction’, a brief grounding about basics of automatic text visualization, its challenges with a ground on proposed work. A brief review of exiting related work is been provided in Section-2. For reader’s convenience, it also puts light on technology used in this research. Section-3 ‘Implementation’ has three sub sections. First discusses about suitable approach adopted for Preksha. Second introduces Scene Grammar and third talks about scene synthesis. Section-4 presents a background on Java3D and SceneGraph. Section-4 presents result analysis and future work. Section-5 concludes the paper.

Prior Work
Readers may refer a detailed survey presented by [18] on varied parameters for the work on ATV. Ma [9, 10], ADVT [1], Wordseye [2], text-to-scene[11] and carsim[15] are some of the popular text visualization system. Although major work is done on English and other languages [12], [3], [4], [2] and [15] for creation, representation, structure, and using large computational linguistic resources from knowledge network but no research is found for Hindi language except our work [5] outlines a basic design of application for conversion from a given Hindi input text into an image. Our prior works [6], [7], [8], [13], [16], [20], and [22] that have presented the same research at different levels. This paper further extends our previous work on topic.

Java 3D [21] is an application programming interface (API) developed at Sun Microsystems for rendering interactive 3D graphics using the Java programming language. Java 3D is a client–side Java API. Other examples of Sun client–side APIs include the Abstract Windows Toolkit (AWT) and Java Foundation Classes (JFC/Swing), which are both Java class libraries for building applications with a Graphical User Interface (GUI). Java 3D provides a bridge between the virtual and physical environment by constructing a one-to-one mapping from one space to another, letting activity in one space affect the other. Scene graph in the form of suitable Computer Graphics (CG) technology is chosen to work with scene engine. Scene graph (as mentioned in Sun Microsystems, 1999) [14] is a general data structure commonly used by vector-based graphics, editing applications and modern computer games. It is a robust and extensible commonly used hierarchical data structure where the leaf nodes represent objects to render. It is easy to build and extend. It arranges the logical and spatial representation of a graphical scene in a family tree form. It is organized in an abstract Data structure - directed acyclic graph (DAG) in lisp notation form. A DAG tree contains no loop and each node has a single parent node. VRML is an international standard for describing 3-D shapes and scenery on the World Wide Web. VRML's technology has very broad applicability, including web-based entertainment, distributed visualization, 3-D user interfaces to remote web resources, 3-D collaborative environments, interactive simulations for education, virtual museums, virtual retail spaces, and more. VRML is a key technology shaping the future of the web.

IMPLEMENTATION
This section presents the details on scene grammar and scene implementation strategy after a brief introduction on overall architecture of an ATV research. Here, we take a sample
sentence – ‘कमरे में टेबल पर गमले के पास फोन है’ [There is a phone near a flowerpot on the table in a room] to present an end-to-end process of the research.

3.1 Architecture ATV Research

The language considered for this research is Hindi, so it is expected and anticipated to have a good support of grammatical knowledge of Hindi language. Along with linguistic capability, this research is also supported by repositories of pictures and 3D models of objects for scene generation. Basic architecture of a Preksha ATV application is based on [5] and presented in Figure - 1.

Natural language processing (NLP) techniques take place on input text. This NLP engine constitutes pre-processing, part-of-speech (POS) tagging, Language parsing using Tree adjoining grammar formalism (TAG) [17], semantic role labeling, dependency identification, and anaphora resolution. The acquired knowledge is structured for knowledge representation scheme, as presented in our prior research [7]. The output of the NLP engine is a dependency tree; it populates the information as a structured data in the knowledgebase. The output of knowledge engine helps in generating the final scene instance. The output of NLP engine is a parsed tree presented in Figure 2.
3.2 Scene Planning

Scene grammar is a collection of visualizability semantic rules and 3D models annotated with it. This is the core component for scene planning consequently a final scene synthesis in VRML (‘.wrl’) file format. In parallel, scene-planning task is carried out to knowledge extraction process. This scene plan helps generating virtual environment using scene resources. For Scene grammar, an offline resource repository is prepared with linguistic rules, semantic features, and corresponding 3D object model. Mainly, the 3D models in vrml format are already mapped with their corresponding object’s name and features in linguistic repository. These models in VRML files are tagged with their object with Hindi name, English name, synonyms, grammatical category, and semantic relations. The semantic roles such as background, container, and object from the input text are already identified by the knowledge engine. The placement of objects are more likely in sequence from independent to dependent objects as shown in Figure - 3.

---

**Figure – 2. Language processing output of sentence – “कमरे में टेबल पर गमले के पास फोन है”**

**Figure – 3. Scene plan hierarchy**

Background $\rightarrow$ container $\rightarrow$ (main)object $\rightarrow$ (dependant)object $\rightarrow$ ...$\rightarrow$ (dependant)object $^n$

**Figure – 4. Scene generation engine**
The graphical constraints of dependent objects like position, orientation, size, color, and texture of objects, are obtained from the semantic relations with parent object in the scene. As shown in Figure – 4, there are two basic steps in the scene generation a) scene planning using scene grammar and b) scene synthesis. This scene planning process carries the information of the extracted visually important objects of environment and their attributes. The scene environment is generated by identifying the objects, their inter-relations, and their properties. Transfer rules and adequate heuristics are applied to resolve conflicts and to add implicit constraints in the scene environment to be generated. Collisions of new objects with the reference objects are avoided in the placement algorithm.

### 3.3 Scene Synthesis

Using Scene grammar by extracted knowledge, a scene is synthesized with background preparation, objects selection, spatial relation, merging and positioning, scene placement and final Image rendering. VRML is used for scene processing. A dynamic runtime VRML is prepared using information processed while scene planning. Scene plan is processed by arranging visualizable objects along with and their spatial relations and attributes into VRML nodes. Finally, the prepared VRML composes the virtual environment. VRML file can be viewed in user interface as well as in other open sources software and supported browsers. Here, we talk about preparing a runtime VRML file with file format ‘.wrl’ based on processed scene grammar. A VRML is a set of nodes. Each node will have a type, scale, rotation, translation, color, texture, and some others. A VRML starts with header with version number and encoding utf8 as shown in Figure - 5.

```text
#VRML V2.0 utf8
```

**Figure – 5 Sample of VRML Header**

The second important node is a ‘transform node’, a group node. As a group node, it can be used to define a set of nodes as a single object. This node allows defining a new local coordinate system for the nodes within the group. The hierarchy of ‘transform node’ that consists of translation, rotation, and scale nodes etc. as mentioned in Figure - 6. All the nodes inside a 'transform node' are affected by these transformations, i.e. all coordinates are computed in the local coordinate system. Another ‘transform groups’ inside one ‘transforms group’ accumulate the transformations specified in each Transform. The inner Transform group defines a local coordinate system based on the coordinate system defined in the outer transform group. The fields of ‘transform node’ are presented.

- **children** contains all the nodes included in the group.
- **scale** specifies a 3D scaling transformation.
- **scaleOrientation** defines a rotation of the axes for the scaling operation.
- **center** defines the center of the scaling transform.
- **rotation** defines a rotation on an arbitrary axis.
- **translation** defines the origin of the local coordinate system.
- **bboxCenter** specifies the center of a box that encloses the nodes in the group.
- **bboxSize** specifies where the size of a box that encloses the nodes in the group.

For preparing a complete virtual environment, it is desirable to plan and prepare for background and Meta information apart from event presentation. Here, we discuss on these aspects along with describing their corresponding nodes.
a. **Background preparation**

The background of a scene is an image or gradient colour that is a visual representation of surroundings. The `<background>` tag in knowledge-XML (refer Figure - 3) provides the information about background object, property and associated resource filename. The `<background>` node in VRML helps in preparing desired background. The background node allows defining the sky, ground, and panorama images to add a horizon to your world. Default background of the scene is mentioned with `skyColor`, `skyAngle`, `groundColor` and `groundAngle`. The `<SceneInfo>` tag contains `<background>` sub tag that has `<filename>` sub tag in knowledge-XML. The filename mentioned `<filename>` sub tag is the name of resource file that is to be rendered in virtual environment. This image `<image_name>` of background is added into background node of VRML file as `frontUrl <image_name>`. It is shown in Figure 7.

```
Transform {
  scale 1 1 1
  scaleOrientation 0 0 1 0
  center 0 0 0
  rotation 0 0 1 0
  translation 0 0 0
  bboxCenter 0 0 0
  bboxSize -1 -1 -1
  children [......]
}
```

```
Background {
  skyColor [0.0 0.2 0.7, 0.0 0.5 1.0, 1.0 1.0 1.0]
  skyAngle [1.309, 1.571]
  groundColor [0.1 0.10 0.0, 0.4 0.25 0.2, 0.6 0.60 0.6]
  groundAngle [1.309, 1.571]
  frontUrl "room.jpg"
}
```

Figure – 6. VRML Transform Node

Figure – 7. VRML Background Node example

b. **Object placement**

After processing of background knowledge, information like container and objects are retrieved from knowledge-XML. The mentioned objects in knowledge-XML are inserted using `'Inline node' in 'children node' of 'Transform node'`. It is supported by the default properties of objects like dimensions, color, visualizability, and containability that are already preserved in model-database in a form of static semantic feature. The `'Inline node'` creates a special group that needs URL of wrl file of corresponding object as mentioned. Snippet in Figure 8 shows two type of practice to insert a table object named `table.wrl` and `chair.wrl` into final VRML.

```
#VRML V2.0 utf8
Inline {
  url "table.wrl"
}
Transform {
  translation 3 0 0
  children [
    Inline {
      url "chair.wrl"
    }
  ]
}
```

```
Transform {
  translation -9.0 -6.0 0.0
  children [
    Inline {
      url "table.wrl"
    }
  ]
}
```

Figure – 8. Sample of VRML Inline Node

Figure – 9. Sample of VRML object positioning
c. **Object positioning**

A detailed work on spatial relation is presented in [Spatial] and [preposition]. The ’translation’ attribute in ‘Transform node’ tells about the placement co-ordinates along the X, Y, and Z-axes. VRML is based on a 3-dimensional coordinate system. In VRML, the X-axis goes from left to right, parallel to the bottom of the screen. The Y-axis goes from top to bottom, parallel to the sides of the monitor. The Z-axis goes from the monitor to your nose, and back into the screen. While planning for a scene, it is found logical to place a background of the scene first, than container and then finally the rest of the objects (refer Figure - 4). The position of objects in VRML file is defined using translation field in ‘Transform node’. The syntax for position an object is ’translation x, y, z’ (refer Figure - 9) and the scale used for these axis is in meter/cm. The contained objects get their relative positioning as per their spatial relation to the container.

d. **Object Resizing**

The ’scale’ attribute tells how big the thing should grow in the X, Y, and Z-axes. This transformation allows us to modify an existing shape by specifying a factor to multiply on each axis. The default size of object stored in repository is mentioned as static feature named ‘dimension’. We scaled objects and tagged with their default height, width, and depth of the object. ‘Transform node’ includes a ‘scale node’ to change the size of the new coordinate system. It can enlarge or decrease the size of a shape in any number of dimensions. The scale factors must be positive. Syntax is shown in Figure 10. We rescale objects as per the requirement of input attribute like ‘छोटा [small]’, ‘बड़ा [big]’, ‘लम्बा [tall]’ and ‘मोटा [fat]’.

It is worth to mention here that, after rescaling the objects, new dimensions of objects are maintained for reference positioning of other related objects.

![Figure 10. Sample of Object resizing](image)

![Figure 11. Sample of Object Texture](image)

e. **Object Texture**

An ‘imageTexture node’ in VRML makes all of its children have a specific picture for a texture. Input to the node is URL, and direction to tile the picture horizontally and vertically. Node Transform field with texture tag is used to specific color and texture of the objects like – लकड़ी की टेबल [wooden table], सुनहरा बर्तन [golden utensil] or ईंट की दीवार [brick wall] (Figure - 11).
f. Object Color

In case of color attribute, it is not practical to store all the objects in all possible color’s values. So, a provision is made to add node named ‘DirectionalLight’ for color feature. It is used to provide specific color of light impression on that particular model. In this way, a table with red color light may be supplied in absence of actual ‘red table’ model. The color model in VRML is RGB. In order to define a color three values are needed: Red Green and Blue, these values are between 0.0 and 1.0. The ‘DirectionalLight’ node has the basic light fields of intensity, color, and ambient Intensity. It also has an on field, which is simply ‘TRUE’ or ‘FALSE’, indicating whether the light is ‘on’ or ‘off’. It also has a direction field, which is a vector corresponding to the direction that the light shines in. The default value of 0 0 -1, for instance, indicates that the light shines from +Z towards the origin, or down the -Z axis. The full definition of the DirectionalLight node is shown in Figure 12.

![Figure 12. Sample of Directional Light](image)

```
DirectionalLight {
    exposedField SFFloat ambientIntensity 0
    exposedField SFColor color 1 1 1
    exposedField SFVec3f direction 0 0 -1
    exposedField SFFloat intensity 1
    exposedField SFBool on TRUE
}
```

Result analysis and Future work

This section discusses on the results of the proposed framework.

![Figure 13. VRML generation for input text: “कमरे में टेबल पर गमले के पास फोन है.”](image)

![Figure 14. Final scene synthesis for input text: “कमरे में टेबल पर गमले के पास फोन है.”](image)
A design is proposed for Scene Grammar for automatic generation of 3D Scene. A systematic procedure for proposed concept is explained with a brief background of ATV system, its challenges, and overall architecture. It put a light on prior work and technology used for proposed research. After an introduction of scene generation process, it detailed about scene grammar and scene synthesis process. Scene grammar is presented which is the base of scene synthesis process. A Step-by-step Scene Planning is explained in detail with Illustrative examples. Result of work is presented with a discussion on future work. Currently, the proposed work is limited to simple small sentences, whereas the proposed design has scope and extendibility to further enhancement for complex and longer sentences. The applications of proposed work are considered in information visualization, scientific language learning, gaming, virtual reality, and localization.

REFERENCE


VRML, VRML: https://www.w3.org/MarkUp/VRML/. (Accessed on 02 Feb, 2018)

