Automatic Conversion of Traffic Accident Reports into 3D Animations

Richard Johansson*and Pierre Nugues †
Department of Computer Science
Lund University, Sweden

Abstract

This paper describes a system that automatically converts narratives into 3D scenes. The texts, written in Swedish, describe road accidents. One of the key features of the program is that it animates the generated scene using temporal relations between the events. We believe that this system is the first text-to-scene converter that is not restricted to invented narratives.

The system consists of three modules: natural language interpretation based on information extraction (IE) methods, a planning module that produces a geometric description of the accident, and finally a visualization module that uses Java3D to render the geometric description as animated graphics.

We performed a small user study to evaluate the quality of the visualization. The results validate our choice of methods, and since this is the first evaluation of a text-to-scene conversion system, they also provide a baseline for further studies.

CR Categories: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Animations; I.2.7 [Artificial Intelligence]: Natural Language Processing—Text Analysis; I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search—Plan execution, formation, and generation

Keywords: text-to-scene conversion, 3D graphics, natural language processing

1 Introduction

Automatic text-to-scene conversion consists in synthesizing a 2D or 3D geometric description from a text and in displaying it. The scene can be static or animated. Animated 3D graphics have some advantages for the visualization of information. They can reproduce a real scene more accurately and render a sequence of events. In addition, when 3D graphics are coupled with virtual environments, they enable users to navigate in a 3D world and interact with objects.

The conversion of natural language texts into graphics has been investigated in a few projects. NALIG [Adorni et al. 1984] is an early example of them that was aimed at recreating static 2D scenes. One of the major goals of the project was to study relationships between space and prepositions. NALIG considered simple phrases in Italian of the type subject, preposition, object that in spite of their simplicity can have ambiguous interpretations. From what is described in the papers, NALIG has not been extended to process sentences and even less to texts. WordsEye [Coyne and Sproat 2001] is a more recent system that recreates 3D animated scenes from short descriptions. The interpretation of a narrative is based on semantic frames and a good deal of inferences about the environment. WordsEye does not address real world stories. The narratives cited as examples resemble imaginary fairy tales. In addition, all the cited texts appear to have been invented by the authors and not collected from kids, for instance. CogViSys is a last example that started with the idea of generating texts from a sequence of video images. The authors found that it could also be useful to reverse the process and generate synthetic video sequences from texts. The system is limited to the visualization of single vehicle maneuvers at an intersection as the one described in this two-sentence narrative: *A car came from Kriegstrasse. It turned left at the intersection* [Arens et al. 2002]. The authors give no further details on the results.

2 Overview of the Carsim System

The Carsim[1] system [Johansson et al. 2005; Dupuy et al. 2001] is a text-to-scene converter that handles real texts and that we evaluated using quantitative methods. The program generates 3D graphics from traffic accident reports generally collected from web sites of Swedish newspapers. One of its key features is that it takes time and temporal relations between events into account to animate the synthesized scene.

Narratives of a car accidents often make use of space descriptions, movements, and directions that are sometimes difficult to grasp for readers. We believe that forming consistent mental images is necessary to understand them properly. However, some people have difficulties in imagining situations and may need visual aids predesigned by professional analysts.

Carsim tries to address this need. It is intended to be a helpful tool that can enable people to imagine a traffic situation and understand the course of events properly. To generate a 3D scene, Carsim combines natural language processing components and a visualizer. The language processing module adopts an information extraction (IE) strategy and includes machine learning methods to solve coreference, classify predicate/argument structures, and order events temporally.

However, as real texts suffer from underspecification and rarely contain a detailed geometric description of the actions, information extraction alone is insufficient to convert narratives into images automatically. To handle this, Carsim infers implicit information about the environment and the involved entities from key phrases in the text, knowledge about typical traffic situations, and properties of the involved entities. The program uses a visualization planner that applies spatial and temporal reasoning to find the simplest configuration that fits the description.

*e-mail: richard@cs.lth.se
†e-mail:pierre@cs.lth.se

[1]An online demonstration of the system is available at `http://www.lucas.lth.se/lt`.

2.1 A Corpus of Traffic Accident Descriptions in Swedish

Carsim has been developed using a corpus of reports written in Swedish. As development set, we collected approximately 200 reports of road accidents from various newspapers. The task of analyzing the news reports is made more complex by their variability in style and length. The amount of details is overwhelming in some reports, while in others most of the information is implicit. The complexity of the accidents described ranges from simple accidents with only one vehicle to multiple collisions with several participating vehicles.

Although our work has concentrated on the press clippings, we also have a small set of accident reports extracted from the STRADA database (Swedish TRaffic Accident Data Acquisition) of Vägverket, the Swedish traffic authority.

The next text is an excerpt from our test corpus. This report is an example of a press wire describing an accident.

> *Tre personer omkom när en buss och personbil på måndagen kroc*k*ade på väg 55 vid Fornebo i närheten av Flen. Det var ett barn och två vuxna som färdades i personbilen som omkom i olyckan. Ytterligare ett barn, en flicka, fanns i bilen, men kunde ta sig ut. - Hon fick hjälp av en person att ta sig ut ur bilen, berättar Mats Elfwén, räddningsledare vid räddningstjänsten i Flen, för TT. Han vet inte hur olyckan gick till. - Av någon anledning kom personbilen över på fel sida med sladd. Bussföraren försökte undvika den, men det blev en frontalkollision, säger Mats Elfwén. Vid krocken fattade personbilen eld. Flickan som räddades ur bilen fördes till sjukhus med bland annat brännskador. Ungefär 15 personer från räddningstjänsterna i Flen och Malmköping deltog i arbetet vid olyckan.*
>
> Svenska dagbladet, November 11, 2002
>
> Three persons died when a bus and a passenger car collided on Monday on road 55 near Fornebo in the vicinity of Flen. A child and two adults, who traveled in the passenger car, died in the accident. Another child, a girl, was in the car but was able to escape. -She was assisted by a person to get out of the car, reported Mats Elfwén, head of the rescue team in Flen, to the Swedish News Agency. He does not know how the accident occurred. For some reason, the passenger car slid and came over on the wrong side of the road. The bus driver tried to avoid it, but ended in a frontal collision, said Mats Elfwén. During the collision, the passenger car caught fire. The girl that could escape the car was sent to hospital with burns. Approximately 15 persons from the rescue team in Flen and Malmköping participated in the rescue.
>
> The text above, our translation.

2.2 Carsim's Architecture

Carsim's architecture consists of a pipeline of three modules where each module carries out one step of the conversion process (see Figure 1).

- A *natural language processing* module interprets the text to fill a template – an intermediate symbolic representation.
- A *spatio-temporal planning and inference* module produces a full geometric description given the symbolic representation.
- A *graphical* module renders the geometric description as graphics using the Java3D library.

Carsim's language processing module uses information extraction techniques. It reduces the text content to a tabular structure – the template – that outlines what happened. We use this template as an intermediate representation between texts and geometry. This is made necessary because the information expressed by most reports usually has little affinity with a geometric description. Exact and explicit accounts of the world and its physical properties are rarely present. In addition, our vocabulary is finite and discrete, while the set of geometric descriptions is infinite and continuous.

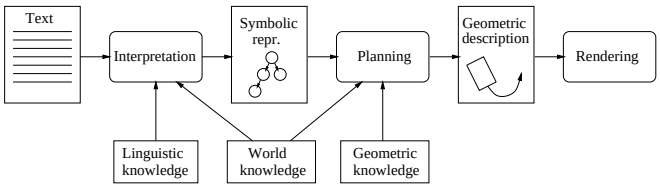

Figure 1: System architecture.

Once the NLP module has interpreted and converted a text, the planner maps the resulting symbolic representation of the world, the entities, and behaviors, onto a complete and unambiguous geometric description in a Euclidean space.

Certain facts are never explicitly stated, but are assumed by the author to be known to the reader. This includes linguistic knowledge, world knowledge (such as traffic regulations and typical behaviors), and geometric knowledge (such as typical sizes of vehicles). The language processing and planning modules take this knowledge into account in order to produce a credible geometric description that can be visualized by the renderer.

2.3 Knowledge Representation

The knowledge representation contained in the template has to manage the following trade-off. In order to be able to describe a scene, it must contain enough information to make it feasible to produce a consistent geometric description, acceptable to the user. On the other hand, the representation has to be close to ways human beings describe things to capture information in the texts.

We used four concept categories that we ordered in an inheritance hierarchy. Each category is implemented as predefined attribute/values slots:

- *Objects*. These are typically the physical entities that are mentioned in the text, but we might also need to present abstract entities as symbols in the scene. Each object has a type that is selected from a predefined, finite set. `Car` and `Bus` are examples of object types.
- *Events*. They correspond intuitively to an activity that goes on during a period in time and here to the possible object behaviors. We represent events as entities with a type from a predefined set. `Impact` and `CatchFire` are examples.
- *Relations and Quantities*. The objects and the events need to be described and related to each other. The most obvious examples of such information are *spatial* information about objects and *temporal* information about events. We should be able to express not only exact quantities, but also qualitative information (by which we mean that only certain fundamental distinctions are made). `Behind`, `FromLeft`, and `During` are examples of spatial and temporal relations.
- *Environment*. The environment of the accident is important for the visualization to be understandable. Significant environmental parameters include light, weather, road conditions,

and type of environment (such as rural or urban). Another important parameter is topography, but we have set it aside since we have no convenient way to express this qualitatively.

2.4 Natural Language Processing

The natural language processing module uses information extraction techniques. It consists of a sequence of components (Figure 2). The first components carry out a shallow parse: part-of-speech tagging, noun phrase chunking, complex word recognition, and clause segmentation. This is followed by a cascade of semantic markup components: named entity recognition, temporal expression detection, object markup and coreference, and predicate argument detection. A temporal component uses verb tenses and other information to infer the temporal structure of the course of events. Finally, the marked-up structures are interpreted, which results in a symbolic representation of the accident.

The development of the IE module has been made more complex by the fact that few tools or annotated corpora are available for Swedish. The only significant external tool we have used is the Granska part-of-speech tagger [Carlberger and Kann 1999].

3 The Planning and Graphical Modules

We use a planner to create the animation from the information extracted by the natural language processing module. It first determines a set of constraints that the animation needs to fulfill. Then, it goes on to find the initial directions and positions. Finally, it uses a search algorithm to find the trajectory layout. This separation into steps does not allow backtracking and introduces a risk of bad choices. However, it reduces the computation load and proved sufficient for the texts we considered, enabling an interactive generation of 3D scenes and a better user experience.

3.1 Setting Up the Constraints

The constraints on the animation are created using the detected events and the spatial and temporal relations combined with the implicit knowledge about the world. The events are expressed as conjunctions of primitive predicates about the objects and their behavior in time. For example, if we state that there is an `Overtake` event where O_1 overtakes O_2, this is translated into the following proposition:

$$\exists t_1, t_2 . \mathit{MovesSideways}(O_1, \mathit{Left}, t_1) \\ \wedge \mathit{Passes}(O_1, O_2, t_2) \wedge t_1 < t_2$$

In addition, other constraints are implied by the events and our knowledge of the world. For example, if O_1 overtakes O_2, we add the constraints that O_1 is initially positioned behind O_2, and that O_1 has the same initial direction as O_2. Other constraints are added due to the non-presence of events, such as

$$\mathit{NoCollide}(O_1, O_2) \equiv \neg \exists t . \mathit{Collides}(O_1, O_2, t)$$

if there is no mentioned collision between O_1 and O_2.

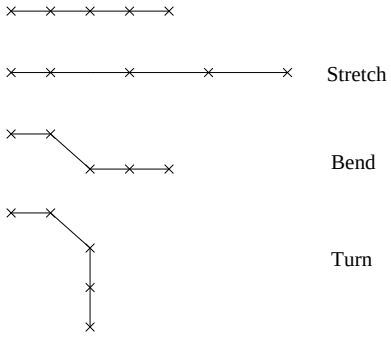

Figure 3: The elementary movements.

3.2 Finding Initial Directions and Positions

We use constraint propagation techniques to infer initial directions and positions for all the involved objects. We first set those directions and positions that are stated explicitly. Each time a direction is uniquely determined, it is set and this change propagates to the sets of available choices of directions for other objects, whose directions have been stated in relation to the first one. When the direction can't be determined uniquely for any object, we pick one object and set its direction. This goes on until the initial directions have been inferred for all objects.

3.3 Finding the Trajectories

After the constraints have been set up, we use the IDA* search method to find a trajectory layout that is as simple as possible while violating no constraints. The trajectories are initially straight, and are modified incrementally until a solution is found. The three types of modifications to the trajectories (i.e. the elementary movements the vehicles can make) are shown in Figure 3. As a heuristic function to guide the search, we use the number of violated constraints multiplied by a scaling constant in order to keep the heuristic admissible.

The most complicated accident in our development corpus contains 8 events, which results in 15 constraints during search, and needs 6 modifications of the trajectories to arrive at a trajectory layout that violates no constraints. This solution is found in a few seconds. Most accidents can be described using only a few constraints.

At times, no solution is found within reasonable time. This typically happens when the IE module has produced incorrect results. In this case, the planner backs off. First, it relaxes some of the temporal constraints (for example: *Simultaneous* constraints are replaced by *NearTime*). Next, all temporal constraints are removed.

3.4 A Planning Example

To illustrate the planning problem, we give an example of a common kind of traffic accident: *A* overtakes *B*, forcing *C* (coming from the opposite direction) off the road. We formalize this using the following constraints:

- $\exists t_1 \mathit{MovesSideways}(A, \mathit{Left}, t_1)$
- $\exists t_2 \mathit{Passes}(A, B, t_2)$
- $\exists t_3 \mathit{LeavesRoad}(C, t_3)$

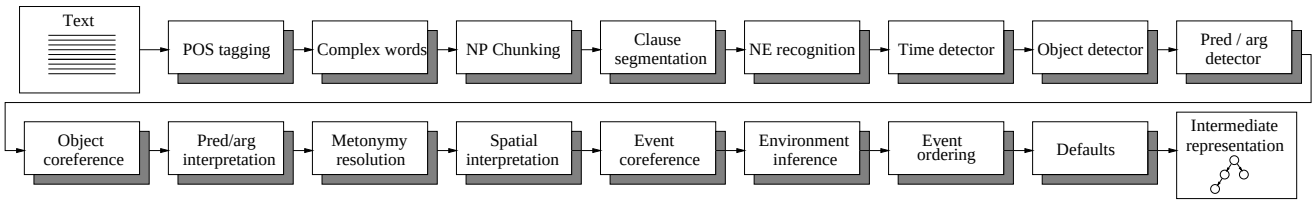

Figure 2: Architecture of the language interpretation module.

- $t_2 > t_1$
- $t_3 > t_2$
- $NoCollide(A,B)$
- $NoCollide(A,C)$
- $NoCollide(B,C)$

We start from initial trajectories, and the search procedure modifies them incrementally. A minimal solution is $Bend(A)$, $Stretch(A)$, $Bend(C)$, meaning that vehicles A and C perform sideways movements, and that vehicle A accelerates (while overtaking).

3.5 Rendering the Geometric Description

We use the Java3D library to render the geometric description as 3D graphics. Java3D is a more or less platform-independent 3D graphics library on top of platform-dependent low-level libraries like OpenGL or DirectX.

Most objects are described using VRML models, although some are implemented using procedural techniques (fires, for example).

Figures 4 and 5 show an example corresponding to the text from Subsection 2.1.

4 Evaluation of the Visualization

To evaluate the system, we used 50 previously unseen texts, which had been collected from newspaper sources on the web. The size of the texts ranged from 36 to 541 tokens. Four users were shown the animations of subsets of the 50 test texts.

The users graded the quality of animations using the following scores: 0 for wrong, 1 for "more or less" correct, and 2 for perfect. The average score was 0.91. The number of texts that had an average score of 2 was 14 (28 percent), and the number of texts with an average score of at least 1 was 28 (56 percent). These figures demonstrate that the chosen strategy is viable, especially in a restricted context like the traffic accident domain. However, interpretation of the figures is difficult since there are no previously published results. In any case, they provide a baseline for further studies, possibly in another domain.

To determine whether the small size of our test group introduced a risk of invalid results, we calculated the standard deviation of annotations[2], and we obtained the value of 0.45. Replacing all annotations with random values from the same probability distribution resulted in a standard deviation of 0.83 on average. In addition, the pairwise correlation of the annotations is 0.75. This suggests that the agreement among annotators is enough for the figures to be relevant.

During discussions with users, we had a number of unexpected opinions about the visualizations. One important example of this is the implicit information they infer from reading the texts. For example, given a short description of a crash in an urban environment, one user imagined a collision of two moving vehicles at an intersection, while another user interpreted it as a collision between a moving and a parked car.

This user response shows that the task of imagining a situation is difficult for humans as well as for machines. Furthermore, while some users have suggested that we improve the realism (for example, the physical behavior of the objects), discussions generally made it clear that the semi-realistic graphics that we use (see Figures 4 and 5) may suggest to the user that the system knows more than it actually does. Since the system visualizes symbolic information, it may actually be more appropriate to present the graphics in a more "abstract" manner that reflects this better, for example via symbolic signs in the scene.

5 Conclusion and Perspectives

We have presented system based on information extraction techniques and symbolic visualization that converts real texts into 3D scenes. It creates animated graphics by taking into account temporal relations between events mentioned in a text and using a planner.

We have provided a quantitative evaluation of a text-to-scene conversion system, which shows promising results that validate our choice of methods and set a baseline for future improvements. As far as we know, Carsim is the only text-to-scene conversion system that has been developed and evaluated using noninvented narratives.

In the future, we would like to extend the system to deeper levels of semantic processing. While the current prototype uses no external knowledge, we would like to integrate additional knowledge sources in order to make the visualization more realistic. An important example of this is geographical and vehicle information, which could be helpful in improving the realism and in creating a more accurate reconstruction of the environment and animation. Another topic we would like to address would be to merge a set of narratives describing a same accident into a unique 3D scene as the animations manually produced by the National Transportation Safety Board (NTSB) in the United States[3].

[2] We calculated this using the formula $\sqrt{\frac{\sum(x_{ij}-\dot{x}_i)^2}{\sum(n_i-1)}}$, where x_{ij} is the score assigned by annotator j on text i, $\dot{x}_i$ the average score on text i, and n_i the number of annotators on text i.

[3] See for instance www.ntsb.gov/events/2000/central_bridge/cb_video.htm

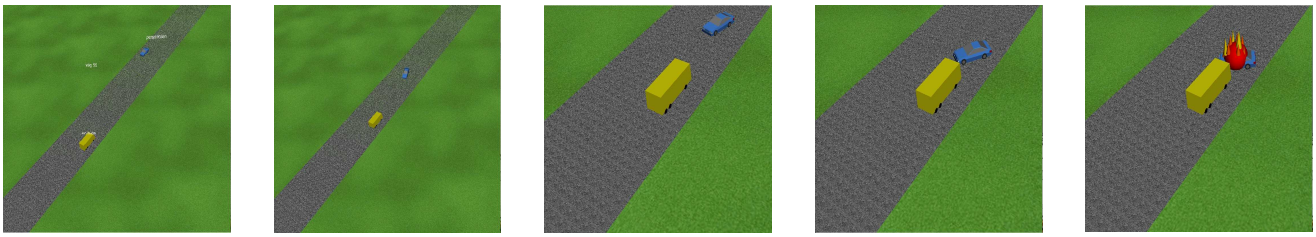

Figure 4: Screenshots from the animation of the text above.

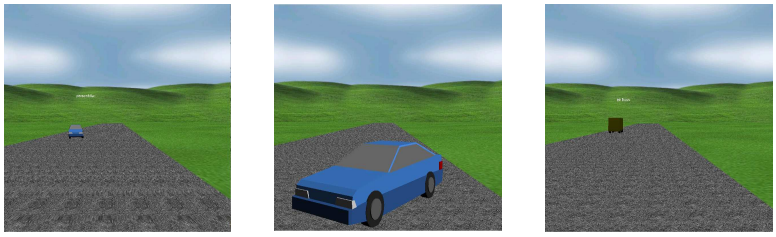

Figure 5: Points of view from the bus and the car.

Acknowledgements

This work was partly supported by grant number 2002-02380 from the Språkteknologi program of Vinnova, the Swedish Agency of Innovation Systems.

References

ADORNI, G., MANZO, M. D., AND GIUNCHIGLIA, F. 1984. Natural language driven image generation. In *Proceedings of COLING 84*, 495–500.

ARENS, M., OTTLIK, A., AND NAGEL, H.-H. 2002. Natural language texts for a cognitive vision system. In *ECAI2002, Proceedings of the 15th European Conference on Artificial Intelligence*, F. van Harmelen, Ed.

CARLBERGER, J., AND KANN, V. 1999. Implementing an efficient part-of-speech tagger. *Software Practice and Experience 29*, 815–832.

COYNE, B., AND SPROAT, R. 2001. WordsEye: An automatic text-to-scene conversion system. In *Proceedings of the Siggraph Conference*.

DUPUY, S., EGGES, A., LEGENDRE, V., AND NUGUES, P. 2001. Generating a 3D simulation of a car accident from a written description in natural language: The Carsim system. In *ACL2001: Workshop on Temporal and Spatial Information Processing*, 1–8.

JOHANSSON, R., BERGLUND, A., DANIELSSON, M., AND NUGUES, P. 2005. Automatic text-to-scene conversion in the traffic accident domain. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, 1073–1078.