# Intelligent Indexing, Querying and Reconstruction of Crime Scene Photographs

Funda Durupınar, Umut Kahramankaptan and Ilyas Cicekli

Department of Computer Engineering
Bilkent University
Ankara, 06800, Turkey

## Abstract

In this paper, we present a system that performs intelligent indexing, querying, and 3-D reconstruction of crime scene photographs from short English descriptions. In our system, natural language is used to describe crime scene photographs. This approach makes use of the object properties and spatial relations between objects in the scene for indexing and scene construction purposes. For retrieval and querying, similarity scores between the extracted relations are calculated. The most important innovation of our research is that it creates a conceptual similarity between spatial relations and it weighs the similarity of arguments with values between 0 and 1. Another development is in defining the inverse operation between relations and the role changing operators with their weights between arguments. Thus, our method calculates the similarity between triplets not only with each other but also with the new triplets that have been created by these new operators.

**Keywords:** Image Indexing, Multimodal Documentation, Image Retrieval, Text-to-Scene Conversion, Scene Construction, Information Extraction

## 1 Introduction

Preservation of the scene after a crime is almost always impossible. Therefore, in order to prevent the destruction of evidence and because photographs capture important information about the event, crime scene officers take various photographs of the scene with varying levels of detail. Then, each of these photographs are indexed and numbered in sequence as Pastra *et al.* [7] have performed. However, retrieval of information from a series of photographs is not easy. At this point, natural language processing (NLP) is an effective medium for the indexing and retrieval of these photographs. In addition, instead of having only 2-D information such as photographs of the event, it would be more convenient for the investigators to examine the crime scene interactively. However, since the original scene cannot be preserved, a virtual scene can be helpful for the investigation process. Here, computer graphics and the creation of a 3-D scene are used. Again, NLP makes the 3-D scene creation an uncomplicated procedure for the user. Thus, a text-to-scene conversion system is used for synthesizing the scene from photograph descriptions.

This paper explains a system that performs indexing, retrieval and 3-D reconstruction of crime scene photographs. Organization of the paper is as follows: Section 2 gives information about similar research areas such as crime scene investigation through NLP and text-to-scene conversion systems. Section 3 describes the methods adopted in this study. Finally, Section 4 gives a discussion about the project, including future work and conclusions.

## 2 Background

Our work is based on the Scene of Crime Information System (SOCIS) [7-10], which can be regarded as an alternative to image retrieval systems as it uses vision-specific features of an image and uses the terms in the description of the image. SOCIS can extract specialist terms and organize them in a conceptual hierarchy and it can identify meaning-bearing relations among the objects depicted in the image. SOCIS integrates an image database with an organized collection of diverse texts that are informative about the images in various ways. Data mined from these texts forms the basis for indexing and retrieval of images.

The SOCIS indexing prototype is a knowledge-based system, where the input is a single caption or a set of captions in plain text. SOCIS takes the captions as input, processes them and extracts relational facts of the form as *class1:argument1 RELATION class2:argument2*, where relations are spatial. Captions express these relations through prepositions and space-denoting verb forms. For the retrieval of the appropriate photographs, SOCIS processes a user query by extracting triplets from it, matching the query triplets with the indexing triplets, and then presenting the matching photographs to the user. In the matching phase, if exact argument matching fails, the program tries to find matches to the semantic expansion of the arguments through the class information. If it still fails, it performs simple argument matching with semantic broadening where applicable.

In our study, we reconstruct the scene by using 3-D computer graphics, which distinguishes our program from SOCIS. The most important project done on text-to-scene conversion is the WordsEye system [4]. This system does automatic conversion of English text into representative three-dimensional scenes. WordsEye first does syntactic and semantic analysis of the input text and then converts the semantic information into low-level depictors that represent 3-D objects, poses, spatial relations, color attributes by means of depiction rules. Syntactic analysis is through the tagging and parsing of the input text. After parsing, the output of the parser is converted into a dependency structure, which is then semantically interpreted and converted into a semantic representation. For linguistic analysis, WordsEye uses existing components such as Church's [2] part of speech tagger, Collins' statistical parser [3] and WordNet [5]. For the scene construction, WordsEye makes use of 3-D polygonal objects. The scene generated is static and does not include animation.

Another application of text-to-scene conversion is CarSim [1,12], which analyzes descriptions of car accidents written in English and constructs the 3-D scenes of these accidents. In CarSim, the resulting scene can be animated as well as being static. Similar to other text-to-scene conversion applications, CarSim has two modules: one for linguistic analysis and another for visualization. In the linguistic analysis, the WordNet lexical database , and the Link-Grammar dependency parser [11] are used. For visualization, static objects are positioned and vehicle motions are planned first. Then, vehicle trajectories are generated. Finally, the scene is rendered and animated accordingly.

## 3 Method

Our system consists of linguistic components and a computer graphics component. In the system, we first get individual sentences from photograph descriptions in English. Then, we send each sentence to the parser and retrieve information from that sentence. The resulting information can be used by both the query and the scene generation systems. Figure 1 shows the design of our system.

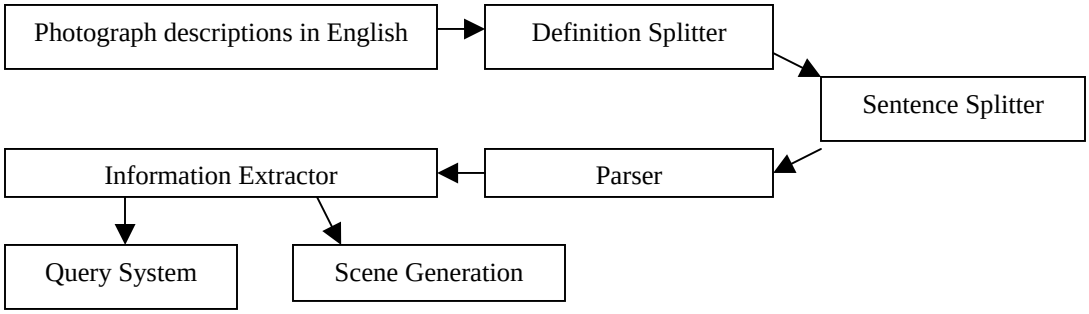



**Figure 1.** System design

### *3.1 Parsing*

In order to extract information from the text, it should be syntactically analyzed at first. For this purpose, syntactic parsing of the text is required. However, not every parser would be effective. We need to retrieve information about objects in the scene. Thus, relationships between words in a sentence must be obtained. Therefore, in our system, we have used a dependency parser. Instead of writing a parser from scratch, as it would be a time-consuming task unrelated to the purpose of the project, we have used Link Grammar Parser to analyze English Sentences [6]. Link Grammar includes an application program interface (API), which enables access to the crucial functions of Link Grammar. Link Grammar has over 60,000 word definitions making up the grammar in the dictionary and it parses one sentence at a time. A sentence is the API's representation of an input string, tokenized and interpreted according to a specific dictionary. The sentences are obtained through the sentence splitter. After a sentence is created and parsed, a set of linkages is returned. A linkage is a sentence with a collection of links. There may be more than one linkages of a sentence. In addition, if the parse has a conjunction, then the linkage is made up of at least two sublinkages.

The output of Link Grammar comes in two forms: The first one is the dependency links between words of a sentence and the second one is a constituent tree. Our concern is on the dependency links rather than the constituent tree. Table 1 gives the connectors used by Link Grammar and utilized in our system.In general, a connector may only link to another with the same name. In addition, connector subscripts determine how connectors are linked. In addition to linkage information, part of speech tags can be obtained from the parser as well. These tags are indicated by suffixes such as .n for nouns, .v for verbs, .a for adjectives and .e for adverbs. Part of speech information is also used in the information extraction phase.

**Table 1.** Link types in Link Grammar

| |
|---|
| **A:** connects pre-noun ("attributive") adjectives to following nouns. |
| **AN:** connects noun-modifiers to following nouns. |
| **D:** connects determiners to nouns. |
| **J:** connects prepositions to their objects. |
| **M:** connects nouns to various kinds of post-noun modifiers. |
| **O:** connects transitive verbs to their objects, direct or indirect. |
| **P:** connects forms of the verb "be" to various words that can be its complements: prepositions, adjectives, and passive and progressive participles. |
| **S:** connects subject nouns to finite verbs. |

### *3.2 Information Extraction*

The output of the parser module is used in extracting information and constructing semantic representations. This section explains the semantic representations and the rules to create these representations.

#### *3.2.1 Semantic Representations*

There are two types of semantic representations: One for displaying objects with their attributes and the other for displaying spatial relationships between two objects in the scene. The first type of representations is as: *Class:OBJ(name, count,* $s_x$*,* $s_y$*,* $s_z$*,* $c_r$*,* $c_g$*,* $c_b$*, material),* where, *name* is the object's name, *count* is the quantity of that object, and $s_x$, $s_y$, $s_z$ are scaling parameters all in x, y and z directions. $c_r$, $c_g$, $c_b$ determine the red, green and blue color components. *Material* determines what that object is made of. *Class* in the representation of the objects is the most specific ontological concept describing that object in our ontology.

The second type of representations is as:

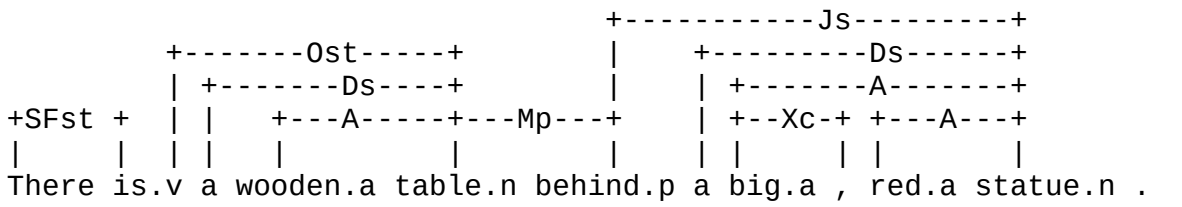
- *ABOVE(object$_1$; object$_2$)*
- *BELOW(object$_1$; object$_2$)*
- *ON(object$_1$; object$_2$)*
- *UNDER(object$_1$; object$_2$)*
- *INSIDE(object$_1$; object$_2$)*
- *AROUND(object$_1$; object$_2$)*
- *RIGHT(object$_1$; object$_2$)*
- *LEFT(object$_1$; object$_2$)*
- *FRONT(object$_1$;object$_2$)*
- *BEHIND(object$_1$; object$_2$)*

where, *object$_1$* and *object$_2$* are the objects that are related to each other spatially.

During the information extraction, we get attributes *quantity*, *size, color* and *material*. *Quantity* attribute of an object is reflected in its *count* parameter. *Size* attribute of an object is reflected in its *scale* parameters. For instance, for the "big" adjective, we scale the object by 2 in all directions, or for the "narrow" adjective, we scale the object in x direction by a factor of 0.5, reducing the width of the object by 50%. All this information is stored in our dictionary. The "size" dictionary includes the adjectives with their scaling factors. *Color* and *material* attributes are reflected in the *color* and *material* parameters. Again, color information is stored in the "color" dictionary and material information is included in the specified "material" dictionary. For instance, the color "yellow" is stored with its corresponding RGB components as (1.0, 1.0, 0.0). In addition, materials are represented as the textures on objects.

#### 3.2.2 Rules to Extract Information

For extracting information, a rule-based approach is adopted. These rules are summarized in Tables 2, 3 and 4. We can see how the rules are applied on an example. For instance, parser output of the sentence "There is a wooden table behind a big, red statue." is as:




First of all, the new objects, i.e. "table" and "statue" are extracted by checking the right hand side of Ds links (Rule 1.a in Table 2). Then, attributes of these objects are retrieved, where *ClassOfTable* is the most

specific ontological concept for *table* and *ClassOfStatue* is the most specific ontological concept for *statue*. Thus, adjective "wooden" is attributed to "table" and adjectives "big" and "red" are attributed to "statue" (Rule 1.a in Table 3).Finally, the 'behind' relationship is extracted by applying Rule 2 in Table 4.

The semantic representation of this sentence is:

*ClassOfTable:OBJ(table, 1,1.0, 1.0, 1.0, 1.0, 1.0, 1.0, wooden).*
*ClassOfStatue:OBJ(statue, 1, 2.0, 2.0, 2.0, 1.0, 0.0, 0.0, nomaterial).*
*BEHIND(ClassOfTable:table; ClassOfStatue:statue),*

where *ClassOfTable* is the most specific ontological concept for *table* and *ClassOfStatue* is the most specific ontological concept for *statue*.

**Table 2.** Extracting new objects

| |
|---|
| 1. For singular count nouns, Ds links are checked for the all sublinkages.<br>a. If left-end od Ds is "a" or "an", new object is inserted.<br>b. Otherwise, no object insertion |
| 2. For plural count nouns, Dmc links are checked for all sublinkages.<br>Left end of a Dmc link denotes a number |
| 3. For mass nouns, POS tags of words are checked and the noun without a Ds link is selected. |

**Table 3.** Extracting the attributes of objects

| |
|---|
| 1. A and AN links are checked for all sublinkages.<br>a. The left-end of A is an attribute of an existing object<br>b. If the left-end is one of left/right/bottom/top/front/back, then append it to the name. |
| 2. Ss links followed by Pa links are checked for all sublinkages.<br>Left-end of Ss is the object; right-end of Pa is the defining adjective. |

**Table 4** Extracting relationships between objects

| For all linkages, |
|---|
| 1. Ss links followed by Pp links followed by Js links are checked.<br>a. Left-end of Ss is the first object, [left-end+1,right-end] of Pp is the preposition, right end of Js is the second object. Such a relation must not have been added before. Also, object1's name should not be the same as object2's name.<br>b. If there is a direction information, append it to the noun before adding the relation ship. |
| 2. Mp links and Js links with right-end of Mp equal to left-end of Js are checked.<br>a. Left-end of Mp is object1<br>b. Right-end of Js is object2<br>c. [left-end+1,right-end] of Mp is the preposition<br>d. If there is a direction information, append it to the noun before adding the relationship |
| 3. Mg links followed by Os links, Ss links followed by Os links, and Ss links followed by Pg*b links followed by Os links are checked.<br>a. Left-end of Mg is object1<br>b. Right-end of Mg is relation verb<br>c. Right-end of Os is object2<br>d. If there is a direction information, append it to the noun before adding the relationship |

### *3.3 Querying*

The system's interface prompts the user to think as if completing a sentence of the form "show me all cases that have photographs in the database that depict...". This query is then processed exactly as if it were a caption. Relational facts are extracted from the query. Then, these relational facts are matched against each photograph's indexing terms and similarity scores are computed. More complicated query structures like querying with more than one description and queries with more structured input may also be possible.

Our database consists of case descriptions in a semi-structured pattern. Descriptions of general scene photographs are in free-form text. These descriptions are marked with non-word strings. First, the text of the general description is stored. The following elements are the photograph description texts. Every description is processed and their spatial information is attached to their description texts.

Our Ontology is derived from WordNet via ad-hoc methods. It has a top-level hierarchy, like WordNet. Its tree structure makes the computation easy. Similarity functions of spatial relationships that will be explained in the next section depend on tree structure and top-level hierarchy properties.

#### 3.3.1 Relation Structure

Relations are triplets and they have a structure as:

$$T(Arg_1;Rel;\ Arg_2) = Rel\ (Class_1 : Arg_1;\ Class_2 : Arg_2)$$

where *Rel* represents the kind of relationship between $Arg_1$ and $Arg_2$. $Arg_1$ and $Arg_2$ represent two objects from scene of crime and $Class_1$ and $Class_2$ represent the ancestors of $Arg_1$ and $Arg_2$ in the ontology relatively.

In order to measure the similarity of two triplets, it is necessary to measure the similarity of all the main structures making up the triplet. For instance, $T_1 = (Class_1 : Arg_1;Rel;Class_2 : Arg_2)$ has the main structures as $Class_1$, $Class_2$, $Arg_1$, $Arg_2$ and *Rel* .

The semantic distance (or semantic similarity) between two objects $Class_1, Class_3$ of two relations
$T_1 = (Class_1 : Arg_1;Rel;Class_2 : Arg_2)$ and $T_2 = (Class_3 : Arg_3;Rel;Class_4 : Arg_4)$
can be calculated using Ontology, which is a tree, as:

$$OntoSim(Class_1,Class_3) = 1/(1+length(ShortestPath(Ontology\ Tree,\ Class_1,Class_3))$$

Similarity between two objects $Arg_1, Arg_3$ with their properties can be calculated with the formula *ArgSim*

$$ArgSim(A,B) = \beta_1 * Match(Arg_{1Name}, Arg_{3Name}) + \beta_2 * Match(Arg_{1Count}, Arg_{3Count}) + \beta_3 * Match(Arg_{1Adj}, Arg_{3Adj})$$

$$Match(X,Y) = \gamma_1 * Equal(X_{Scale}, Y_{Scale}) + \gamma_2 * Equal(X_{Color}, Y_{Color}) + \gamma_3 * Equal(X_{Material}, Y_{Material}) \quad \text{if X and Y are in Adjective Type}$$

$$Match(X,Y) = Equal(X, Y) \quad \text{Otherwise}$$

$$Equal(X, X) = 1$$

$$Equal(X, Y) = 0 \quad \text{if } X \neq Y$$

The weights $\alpha_i$, $\gamma_j$, $\beta_k$ may be identified experimentally.

We should take advantage of the similarity between relations in the same manner as we make use of the similarity between objects for the ontology. Relations may have opposite meanings to each other. They may point at similar locations around the object as spatial relations, like right and left or right and front. Thus, in order to find the similarity ratios between triplets, we can exploit the conceptual similarity between their relations. Similarities between two relations are represented in Table 5. These values are assigned intuitively.

For a basic similarity check between two triplets $T_1 = (Class_1 : Arg_1; Rel; Class_2 : Arg_2)$ and $T_2 = (Class_3 : Arg_3; Rel; Class_4 : Arg_4)$, the following equation is calculated.

$$BasicSim(T_1; T_2) = Sim(R_1,R_2) * ( \alpha_1 * OntoSim(Class_1,Class_3) + \alpha_1 * OntoSim(Class_2,Class_4) + \alpha_2 * ArgSim(Arg_1,Arg_3) + \alpha_2 * ArgSim(Arg_2,Arg_4) ),$$

$$Sim(T_1; T_2) = \text{MAX } \{BasicSim(T_1;T_2), BasicSim(T_1;\mathbf{C}T_2), BasicSim(\mathbf{CI}T_1;T_2), BasicSim(\mathbf{CI}T_1; \mathbf{C}T_2)\}$$

$\mathbf{C}T_1 := (Class_2 : Arg_2; Rel; Class_1 : Arg_1)$
$\mathbf{I}T_1 := (Class_2 : Arg_2; Rel^{-1}; Class_1 : Arg_1)$

The value of $Rel^{-1}$ can be found in Table 6.

- Inverse operator ***I*** is used for representing information in a different way. E.g.

  $ABOVE(a,b)=ABOVE^{-1}(b,a)=BELOW(b,a).$
  $\mathbf{I}\ ABOVE(a,b) = BELOW(b,a).$

  In order to refrain from giving penalty to the opposite relations, the following equation must be used since the inverse operator does not change the information.

  $$Sim(\mathbf{I}\ T_1; T_2) = Sim(T_1; T_2)$$

- Change operator ***C*** is used for representing mirror information.

  $\mathbf{C}\ ABOVE(a,b) = ABOVE(b,a).$

  Here, only the variable positions are changed. This is very similar information but different in the sense of parameter positions. This can be helpful to find similar cases when used with the original information. Thus, the following equation must hold.

  $$Sim(\mathbf{C}T_1;T_2)= PUNISHMENT\_VALUE * Sim(T_1; T_2)$$

Classification of the combinations in the set created by using ***C*** and ***I*** operators due to *BasicSim* values is as follows:

$$\{BasicSim(T_1;T_2), BasicSim(T_1;\mathbf{C}T_2), BasicSim(\mathbf{CI}_T1;T_2), BasicSim(\mathbf{CI}T_1; \mathbf{C}T_2) \}$$

With an optimistic approach, the maximum of these values can be used as a similarity value. Taking maximum can be helpful to prevent a semantic error in the extraction of that relation.

When more than one relational fact is extracted from the query, the system attempts to match each query triplet with each indexing term of each photograph and a sum of the scores that each photograph receives is calculated and used for the final selection of the most appropriate images to be returned to the user. In

cases where no relational facts can be extracted from the query, no simple keyword extraction is done. The similarity between a query description and a photograph description is found as follows:

***Function*** *findSimilarityDesc(QueryDescription $D_1$, PhotographDescription $D_2$) {*

$$maximized\_similarity = \sum_{\mathrm{Re}l_1 \in D_1.\mathrm{Re}lations} find\_\max\_similarity(\mathrm{Re}l_1, D_2)$$

***return*** *maximized_similarity*

*}*

***Function*** *find_max_similarity(Relation Triplet R, Description D) {*

*$\exists Rel_1 \in D.Relations \ \forall Rel_2 \in D.Relations$ such that $Rel_1 \neq Rel_2$* ***and*** *$Sim(Rel_1,R) \geq Sim(Rel_2,R)$*

***return*** *$Sim(Rel_1,R)$*

*}*

**Table 5.** Similarity values between spatial relationships

| $Sim(R_1,R_2)$ | ABOVE | BELOW | AROUND | RIGHT | LEFT | FRONT | BEHIND | IN | ON |
|---|---|---|---|---|---|---|---|---|---|
| **ABOVE** | 1 | 0.7 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.15 | 0.1 |
| **BELOW** | 0.7 | 1 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.15 | 0.1 |
| **AROUND** | 0.2 | 0.2 | 1 | 0.5 | 0.5 | 0.5 | 0.5 | 0.7 | 0.1 |
| **RIGHT** | 0.2 | 0.2 | 0.5 | 1 | 0.7 | 0.2 | 0.2 | 0.15 | 0.1 |
| **LEFT** | 0.2 | 0.2 | 0.5 | 0.7 | 1 | 0.2 | 0.2 | 0.15 | 0.1 |
| **FRONT** | 0.2 | 0.2 | 0.5 | 0.2 | 0.2 | 1 | 0.7 | 0.15 | 0.1 |
| **BEHIND** | 0.2 | 0.2 | 0.5 | 0.2 | 0.2 | 0.7 | 1 | 0.15 | 0.1 |
| **IN** | 0.15 | 0.15 | 0.7 | 0.15 | 0.15 | 0.15 | 0.15 | 1 | 0.1 |
| **ON** | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 1 |

**Table 6.** Reverse operation defined between spatial relationships

| **R** | ABOVE | BELOW | AROUND | RIGHT | LEFT | FRONT | BEHIND | IN | ON |
|---|---|---|---|---|---|---|---|---|---|
| $\mathbf{R^{-1}}$ | BELOW | ABOVE | IN | LEFT | RIGHT | BEHIND | FRONT | AROUND | ON |

### *3.4 Scene Generation*

In the scene generation, initially, the objects, which are polygonal, are rendered due to their attributes. Objects are stored as 3-D polygonal objects. If object count is more than one, it is depicted as one object, not many objects.

For the spatial relationships, bounding boxes of objects are calculated. By default, positive y direction is the upward direction, negative x is the left, positive z is the front direction. Thus, *ABOVE, BELOW, UNDER, RIGHT, LEFT, FRONT* and *BEHIND* relations are implemented by translating the objects due to the amount calculated from their bounding boxes. In addition, *INSIDE* and *AROUND* relations are implemented by overlapping the center of the bounding boxes of the two objects. Computing these relationships are all straightforward. However, calculating the *ON* relationship is not so simple since there is no previously defined direction for the surface of an object to be considered. For instance, a table has the positive y direction for an "on" surface, whereas a wall may have the negative x direction for its 'on' surface. Thus, we have defined "on" normals for all the objects. Then, whenever the relationship *ON($object_1$, $object_2$)* is encountered, certain transformations are done on *$object_1$* to overlap its surface normal with that of *$object_2$*'s.

The transformation values for all these relations can be shown as:

***ABOVE(object$_1$; object$_2$):*** *object$_1$.translate.y := boundingbox$_1$.ymax – boundingbox$_2$.ymin + 1*
***BELOW(object$_1$; object$_2$):*** *object$_1$.translate.y:= boundingbox$_2$.ymin – boundingbox$_1$.ymax – 1*
***RIGHT(object$_1$; object$_2$):*** *object$_1$.translate.x := boundingbox$_1$.xmax – boundingbox$_2$.xmin*
***LEFT(object$_1$; object$_2$):*** *object$_1$.translate.x := boundingbox$_2$.xmin – boundingbox$_1$.xmax*
***FRONT(object$_1$; object$_2$):*** *object$_1$.translate.z := boundingbox$_2$.zmax – boundingbox$_1$.zmin*
***BEHIND(object$_1$; object$_2$):*** *object$_1$.translate.z := boundingbox$_2$.zmin – boundingbox$_1$.zmax*
***INSIDE(object$_1$; object$_2$):*** *object$_1$.translate := boundingbox$_2$.center – boundingbox$_1$.center*
***AROUND(object$_1$; object$_2$):*** *object$_1$.translate := boundingbox$_1$.center – boundingbox$_2$.center*

***ON(object1, object2)***

1. *($\alpha$,direction)=CalculateRotationAngle(on_normal(object$_2$), on_normal(object$_1$))*
2. *object$_1$.rotate.direction.angle = $\alpha$*
3. *object$_1$.translate:= boundingbox$_2$.center – boundingbox$_1$.center*
4. *Move object$_1$ to the surface of object$_2$*

Figure 2 shows an example scene from the text : “ There is a carpet on the floor and a door on the front wall. A window is on the right wall. There is a painting on the left wall. “

**Figure 2.** The viewing area with a generated scene

## 4 Conclusions & Future Work

Our system is a practical application that uses natural language processing techniques combined with 3-D computer graphics. The success of our program mainly depends on the parser since other components all use the output of the parser. However, since similar studies are not very common, it is virtually difficult to compare the success rate of our program with other systems. In addition, access to public data is restricted. Therefore, our main problem in this project can be considered the collection of data. In its current state, our project can be considered just an initial step towards our objectives; however, there are many possibilities for improvement.

For future work, a specialized corpus on crime scene descriptions can be obtained to create a specialized dictionary and ontology for this program. For this purpose, our project should be conducted in collaboration with security forces and experts in this area.

Instead of using intuitive values for similarity values between spatial relationships displayed in Table 5, a statistical research or knowledge acquisition from an expert can be done.

In addition, the idea of this project can be enhanced with integrating speech recognition techniques or more advanced rendering facilities into our project. Furthermore, similar research can be done on other expert domains such as architectural design.

## 5 References


1. Akerberg O, Svensson H, Schulz B, Nugues P, “CarSim: An Automatic 3D Text-to-Scene Conversion System Applied to Road Accident Reports” ,*11th Conference of the European Chapter of the Association for Computational Linguistics*, 2003.
2. Church K, “A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text”, *Proceedings of the Second Conference on Applied Natural Language Processing*, 1988.
3. Collins M, “Head-Driven Statistical Models for Natural Language Parsing”, *Ph.D. Thesis*, 1999.
4. Coyne B, Sproat R., “WordsEye: An Automatic Text-to-Scene Conversion System”, *ACM SIGGRAPH* , 2001.
5. Fellbaum C, “WordNet: An Electronic Lexical Database”, *MIT Press, Cambridge*, 1998.
6. http://www.link.cs.cmu.edu/link/
7. Pastra K, Saggion H, Wilks Y, "Intelligent Indexing of Crime-Scene Photographs“, *IEEE Computer Society* , 2003.
8. Pastra K, Saggion H, Wilks Y, "Extracting Relational Facts for Indexing and Retrieval of Crime-Scene Photographs“, *Knowledge-Based Systems, vol. 16 (5-6), pp.313-320, Elsevier Science*, 2002.
9. Pastra K, Saggion H, Wilks Y, ”NLP for Indexing and Retrieval of Captioned Photographs“, *EACL 2003: 143-146*, 2003.
10. Pastra K, Saggion H, Wilks Y, ”Using Natural Language Processing for Semantic Indexing of Scene-of-Crime Photographs”, *Proceedings of CICLing 2003, Lecture Notes in Computer Science, vol. 2588, pp. 526-536, Springer Verlag,* 2003.
11. Sleator D, Temperley D, “Parsing English with a Link Grammar”, *Third International Workshop on Parsing Technologies*, 1993.
12. Svensson H, Akerberg O, “Development and Integration of Linguistic Components for an Automatic Text-to-Scene Conversion System”, *M.S. thesis* ,2002.