UNL as intermediary for generating graphical simulations of NL discourse

Amitabha Mukerjee M.Chandra Sekhar Reddy
Anand Sudhakar Rajesh Rajsekhar
Vikrant Kumar
I.I.T. Kanpur, INDIA
{*amit, chandra, ands, rajraj, vikrantk*}@iitk.ac.in

Abstract

This paper presents results from the ongoing **Virtual Director** project, which creates graphical animations for simple stories set in a given domain. One of the problems faced is that of designing an intermediate representation for the NL, based on which the graphics motions can be disambiguated. In this paper we propose a 2−step model based on UNL for this purpose. Further, we model the camera motion using cinematographic principles.

1 Introduction

This paper is part of the ongoing "Virtual Director" Project[1] which seeks to generate animated graphic sequences from natural language input. In this work we use the Universal Networking Language(UNL) as an intermediate step. UNL is an electronic language for computers. It intermediates understanding among different natural languages. UNL represents sentences in the form of logical expressions, without ambiguity[3].

Virtual Director works by restricting the Natural Language script to a limited domain, for which extensive semantic information such as the shape, appearance, articulation and action capabilities of objects are encoded as part of the dictionary and knowledge base. For example, considering the domain of an indoor set of rooms whose map is known to the system *a priori*, a script may run as follows:

`A man is sitting in the small room at a table at the center of the room.`
[This creates a table and a chair in the "center" of the room. A man is created and shown sitting on the chair at the table].

`The man is reading a book.`

`He walks to a shelf in the big room.`
[Shows the man getting up (using a fixed camera to the side), then turning parallel to the table (from a fixed camera at an angle). Now a path is found for the man to avoid the table and other obstacles. The walking motion is shown by a camera that tracks the man as he walks along this trajectory.]

```
He picks a book from the top shelf.
```

Animating this sequence involves an initial conversion from NL input to UNL expressions, from that expressions identifying the actions, agents, objects et., then an initial analysis of the semantics of the input sentences is done. For sentences involving actions by an agent, this is done by generating expectations associated with the action verb (the head of the sentence) involved. Next, the objects are created using the procedural models in the database, and positioned according to positional descriptions like "near" or "center', which need to be "concretized"[1]. Next the motion is planned for the articulated agent so that it avoid obstacles and uses a suitable gait to reach the destination. During this process, the camera position for depicting the scene are decided based on a suitable cinematographic "idiom"[2]. Some scenes from the resulting animation is shown in figure.1

This paper focuses on UNL to trace generation which is used in generating animation, agent behaviour and graphic/animation aspects of the reconstruction. Therefore we start from a post-linguistic standpoint where the UNL expressions corresponding to input story is given. First, we convert these UNL expressions into trace. After this, the trace is used to perform concretization of the motion planning parameters. Every object in the scene has, associated with itself points which correspond to different neighbourhoods. This is the aspect which requires disambiguation. Once all data is available numerically, motion planning is initiated and with these results, the cinematographic idioms are sequenced and the animation is rendered.

Earlier versions of this work incorporate a static viewpoint, but in scenes involving motion of the agents, possibility of occlusion and cinematographic aesthetics demand a moving camera.

2 Previous Work

Dr.Amitabha Mukherjee as a part of the Virtual Director project has done work in this field at the Indian Institute of Technology, Kanpur.

The aspects of anthropometrics and modeling are discussed by Norman Badler and Stephen Smoliar[7]. Physical aspects such as the dynamics of complex objects are studied by Hoffman and Hopcroft[8]. This will be used to perform character animation.

Norman Badler has developed Jack, a basic human body animation system[7]. An extension of the basic Jack system by Levison et.al[5] is Soda Jack that simulates a soda bar operator. Action planning in the context of search and manipulation of objects is discussed.

Salesin et.al have created a precise language for the otherwise fuzzy process of defining shots and other basic elements of cinematography[2]. Camera control to amplify the impact of the story can be based on this convention.

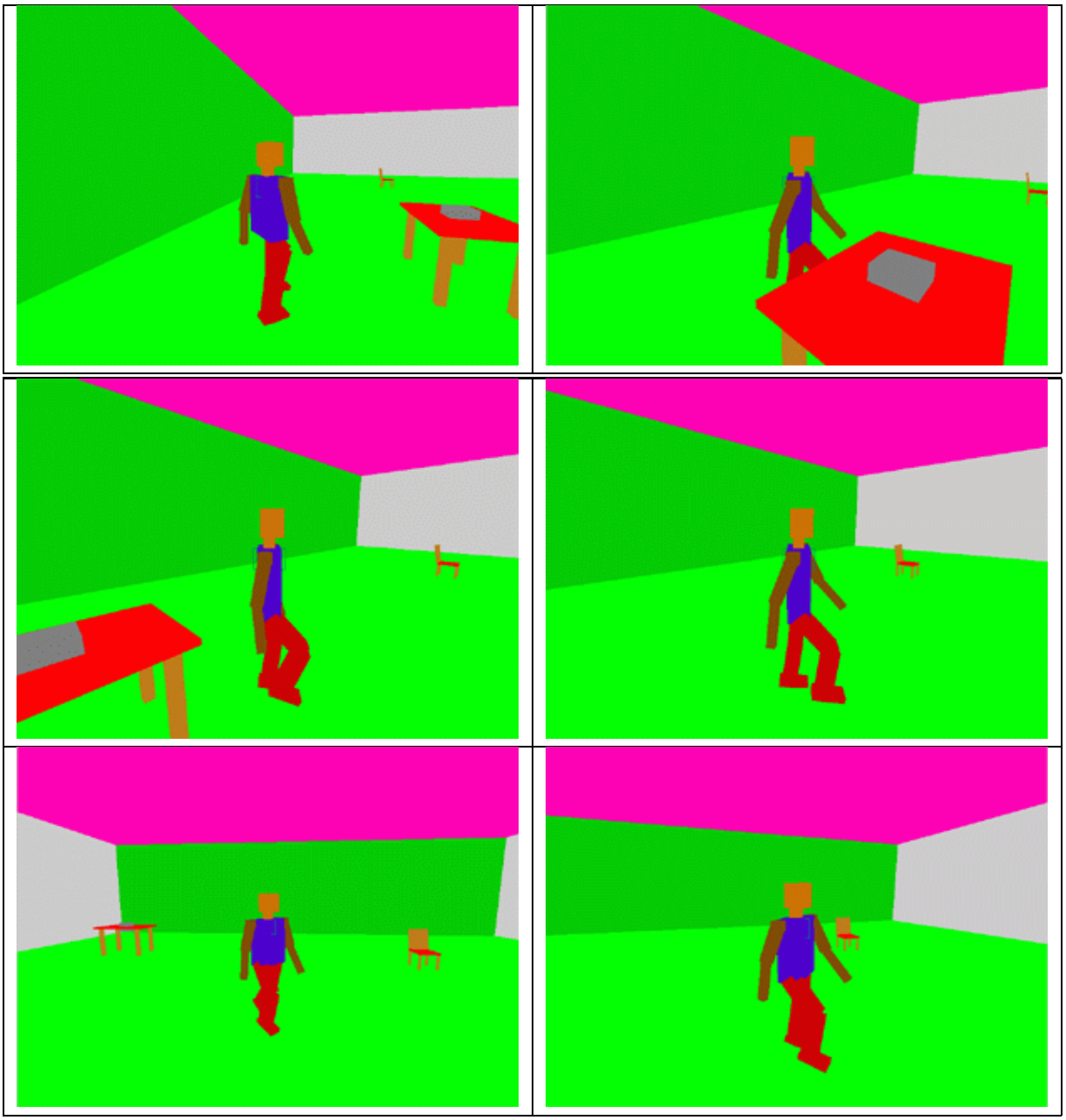


Figure 1: *Animation for a motion sequence.* All object and actor positions are created based on the script. The display uses different cinematographic "idioms" to plan the animation viewpoint. These images were generated for a walk sequence where a dynamic camera movement is desired. The first sequence shows the initial Go-By, the second, third and fourth show the tracking, and the last two images show the final Go-By at the end of the path.

3 Methodology

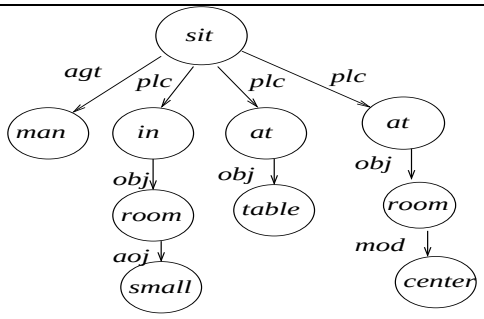
3.1 Conversion into UNL

The UNL consists of Universal Words (UWs), Relations, Attributes, and the UNL Knowledge Base. The Universal Words constitute the vocabulary of the UNL, Relations and Attribute constitute the syntax of the UNL, and the UNL Knowledge Base constitutes the semantics of the UNL[4]. UNL is best suited as intermediary language in generating graphics because through UNL we can easily identify the actions, agents, objects etc, and semantics of the sentences. Moreover it is easy to generate trace of the story through UNL.

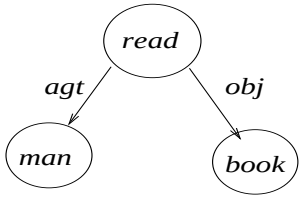
The first step in the generating 3-D animation from the natural language input is converting the NL sentences into UNL representation. We are not handling this part and UNL expressions are given manually. Consider the sentences like `He picks a book from the top shelf` we convert it into UNL

agt(pick(icl>do)@entry@present, he)
obj(pick(icl>do)@entry@present, book(icl>thing))
plc(pick(icl>do)@entry@present, :01)
mod:01(shelf(icl>thing), top)

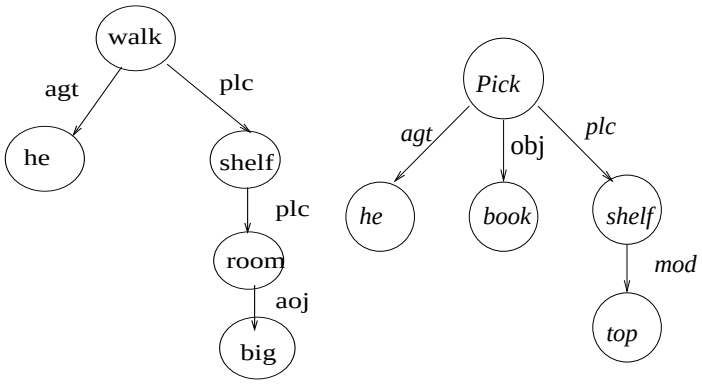
then from these expressions identification of the actions, agents, objects et.are done. Next, the objects are created using the procedural models in the databases, and their positions are determined. In this way we convert all the natural language inputs into UNL expressions then identifying and creation of actions, agents, objects et. is done then performing the specified action which is given in the input story. Following are the UNL Hyper-graphs for story that mention in the introduction.



A man is sitting in the small room at a table at the center of the room.



The man is reading a book



He walks to a shelf in the big roon He picks a book from the top shelf.

3.2 UNL to Trace generation

The UNL expression is not in a form directly usable for camera planning and animation. For this, the UNL expressions are converted into a *trace* which has the property of being numerically precise in terms of the positions and movements of the various entities in the scene. It is a script for animation and camera planning. The stages involves in trace generation are understanding of spatial prepositions, getting containership information, and action decomposition.

3.2.1 Spatial understanding

A small set of spatial prepositions is used to represent information about the position of objects.

Preposition	*Category*
NEAR, BETWEEN, IN	location
RIGHT, LEFT, FRONT, BEHIND	directions

Potential field methods[6] are used to calculate the positions of the objects from the prepositions. The parameters for the potential fields are obtained from the earlier work on human perception of spatial prepositions.

3.2.2 Containership information

When an entity's position is specified in relation to another as being *on* or *under*, that other entity is said to be the *container* of the first. The WALK action requires that the destination not be in the obstacle space. This containership information is used in the action interpolation stage to convert the WALK's destinations to the container

3.2.3 Action decomposition

The trace of the story contains only primitive actions like walk, sit, read, and gaze. So more complex actions have to be broken down into their constituents[5]. This is action interpolation. For example, an action like "jack pickup book" involves a few simple actions. First, jack must *goto* the book. Since the book is probably on a table (container), he will have to walk to the table. Then, jack

must *face* the book. After this, the book has to be picked up. So, the result of interpolation is:

jack goto table
jack face book
jack pickup book

After interpolation, the *goto*s are translated into actual paths in the obstacle space and written in the trace. The trace is composed of blocks, each of which represent a atomic action. The format of a block is:

<Action name>
<list of entities>
*2
<numeric data>
end<crlf>

Action	*Entity list*	*Numeric Data*
WALK	actor, from, to	sequence of points on the path
PICKUP	actor, picked_object	position of actor, position of object
SIT	actor	position
STAND	actor	position
GIVE	giver, taker, exchanged_object	giver position, taker position

Here is the example of trace generation for UNL expressions representing `Jack walks to a shelf`.

UNL expressions are:
agt(walk(icl>do)@entry,Jack)
plc(walk(icl>do)@entry,shelf(icl>thing)).

From these we can identify:
Agent : Jack
Action : walk
Destination Place : shelf

After this we identify their positions:
suppose Jack is at 7.7 and shelf is at 20.17
then calling appropriate procedures for producing numeical data that represents the specified action:
Action name : WALK
Actor : Jack
From : 7.0 7.0
To : 20.0 17.0

Sequence of points on the path:
7.0 7.0
7.2 8.0
7.2 8.1
8.0 8.1
8.9 8.6
9.9 9.4

10.9 10.2
12.0 11.1
14.0 13.0
17.0 16.0
17.8 16.4
20.0 16.4
21.0 17.0
22.0 17.0
end

3.3 Path Planning

For a man to walk between two points, a natural looking path has two requirements. Viz. obstacle avoidance, and error forgiveness and optimality. Since error forgiveness and optimality are two properties which cannot be achieved simultaneously, an optimal path is chosen and then altered to give it error forgiveness.

The obstacles in the domain have, as one of their attributes, their enclosing convex polygon. In the obstacle space they are represented by this convex polygon. Human agents are assumed to be bounded by a circle. The shape of the obstacle in the configuration space is approximated to an enlarged convex polygon with edges parallel to the bounding polygon of the object and at a distance corresponding to the radius of the circle.

Obstacles have a short-range positive potential field. This is used to give the path error forgiveness through repulsion.

A visibility graph is constructed in the configuration space and a path is found from the start to the goal using an A* search[9]. In this initial path, closely spaced (˜1m) intermediate nodes are added and their positions are changed iteratively to incrementally decrease the total energy due to the potential field of the obstacles and the internal energies due to bending and stretching.

The result is a natural-looking, near-optimal, error forgiving path. The addition of intermediate points causes the resulting path to smooth and graceful.

4 Camera Planning Module

At this point, the actual motions are concretized and identified. The camera positions now depend on these positions only.

The need for camera planning arises because of two primary reasons:

1. more aesthetic appeal.

2. highlight the object of interest, determined from the context of the story.

The camera is controlled using the notion of the Idiom, the term being borrowed from the cinematographer's lexicon. The idea behind an idiom is that a story is composed of some primitive actions that can be treated uniformly cinematically as well as from the point of view of animation. This brings homogeneity in the way these activities are treated. So we have idioms for activities like walking, looking, giving, picking, etc..

Cinematographic Idiom	Associated Activity
PATHIDIOM	walking
TALKIDIOM	talking
PICKIDIOM	picking an object, sitting, reading, etc.
LOOKIDIOM	facing or gazing an object:p
OBSTACLEIDIOM	no associated activity

Table 1: The Idioms in relation to the activities.

Therefore to dynamically control the camera it is necessary for the story to be broken down as a list of primitive activities or atomic actions as discussed in the previous section. This is done during trace generation. The trace contains the story broken down as a sequence of these actions or activities which are defined based upon the requirements of the domain. And it turns out that the list is not very long for a sufficient coverage of the domain's activities. Next these actions are to be related to the Idioms that will control the camera positions. The table above provides a list of some of the idioms and their corresponding atomic actions that we have used for our work. As is shown sometimes multiple actions are mapped to the same Idiom. This happens because in these the requirements of the camera placements are the same but they vary in the spatial reasoning and animation modules. So the atomic actions control the way the animation is done and the corresponding Idioms place the camera accordingly. In this way the concept of Idioms not only facilitates camera placement but also simplifies the implementation of our work.

After the trace is received the camera planning proceeds as follows: The trace is read and for each activity it encounters it calls the corresponding Idiom to generate the camera specifications frame by frame. These specifications consist of the camera positions in two dimensional space, the center of viewing cone and the value of the field of view. The Idiom encodes the necessary information required for proper camera movement during that activity. What the Camera Planning Module does is that given the Activity wise breakup of the story it breaks up the activities into the different camera fragment types as dictated by the Idiom governing the activity, and then these fragments are translated into the various camera parameters. The figure below shows the basic camera fragments[2].

A special problem during camera motion planning relates to the situation when the camera following the path specified by the idiom crosses or moves into an extended obstacle like a wall thus blocking the line of sight. In this case since the camera is not doing any obstacle avoidance it doesn't rectify the situation. We propose a solution by adding another Idiom just to handle such contingencies. We call it the ObstacleIdiom, it is just like any other idiom except that it is always active, looking in the background for any indication of obstruction of the line of sight of the camera. As and when any such event occurs it takes control and dictates the path overriding the previous one, but ensuring compatibility with the Idiom that generated the path.

For the ObstacleIdiom we need to distinguish between two situations of camera planning, one involving planning when the actor is dynamic like a walking sequence and one during which the actor is in the same position like sitting or a picking sequence. Now once this distinction is done we need to again distinguish

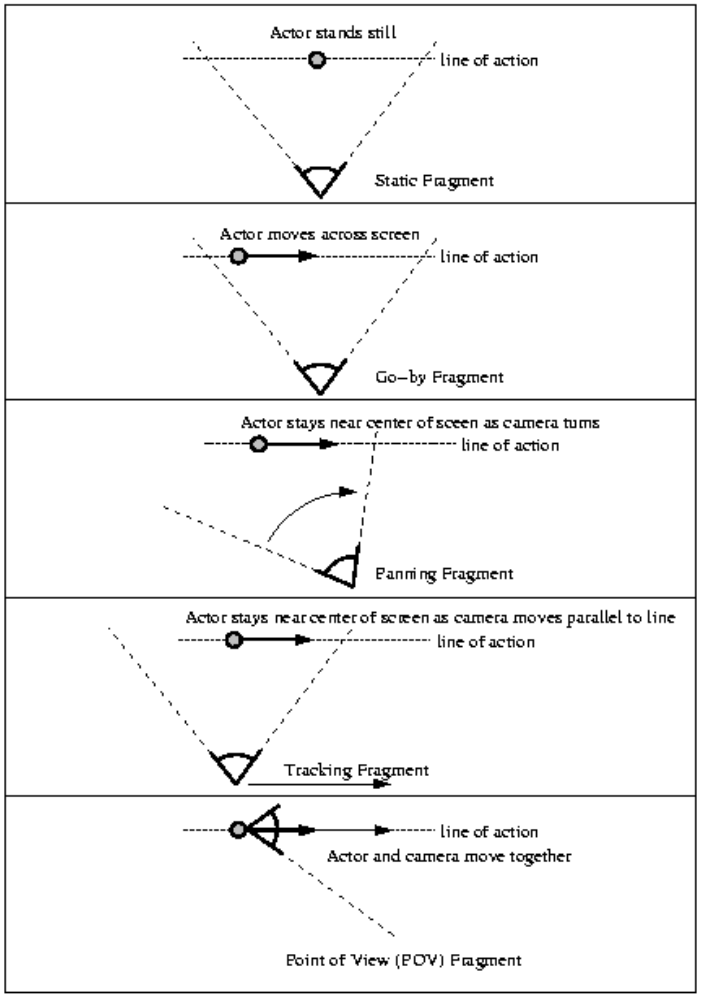


Figure: Fragments are used to specify the type of camera movement. (from Salesin et al. 1996).

between two types of obstacles. To illustrate them consider the following case, a man is moving along a path and an obstacle like a tree comes in between the actor and the camera, the camera will just move past the tree, the tree barely obscuring the man for a moment, the effect even looks good, giving a solid feel of motion. The other case is that of a wall coming in between the camera and the man for an extended period of time, this is something to be avoided. So the two cases of obstacles are

1. momentarily obstructing obstacles like a tree and
2. extended obstacles like a wall.

The distinction is necessary because we identify occlusion by finding if the obstacle lies in between the line segment joining the camera and the actor. The above two cases satisfy the condition but the first case doesn't need any ObstacleIdiom whereas the second one does, so they have to be treated separately.

To avoid the occlusion in case of extended objects a point of view (POV) fragment is used. The figure illustrates the basic idea. The path of the camera passes through a region of the wall, thus occluding the actor for that period when he moves along the path abc, corresponding to points a'b'c' on the path of the camera. Here a' indicates the positon of the camera when the actor is at point a. So the problem is to find an alternate path for the camera instead of a'b'c' such that the actor is visible during this period. For the POV fragment the camera moves along the path of the actor from the behind i.e. it follows the actor from behind for the duration abc.

4.1 Animation

This module receives as input the trace and the camera positions from the previous stages. From this the animation has to be generated. This involves the locomotion of humans, for this we employ key framing. The model of the human body, that we employed at the time of writing of this paper, has fifteen degrees of freedom as follows:

Shoulder	Elbow	Neck	Waist	Hip	Knee	Ankle
3	2	2	3	2	1	2

The gait is modeled through inverse kinematics based on the placement of foot steps along the path.

5 Conclusions

In this paper we have presented two innovations in the field of text-based movie generation. One involves the use of UNL as an intermediate language. This provides for an explicit relational model capturing the agent-object-attribute relations in the natural language input in a standard framework. The second innovation is in the use of cinematographic idiom to improve the presentation of the movie and to highlight the continuity of action and dramatic effects.

In future, we expect to incorporate more discourse specific elements such as tracing individual users and their attributes over the discourse. This will also help resolve anaphora and some of the simpler ambiguities. The trace language used to generate the graphical script may be interaction with other objects in the scene must also be improved.

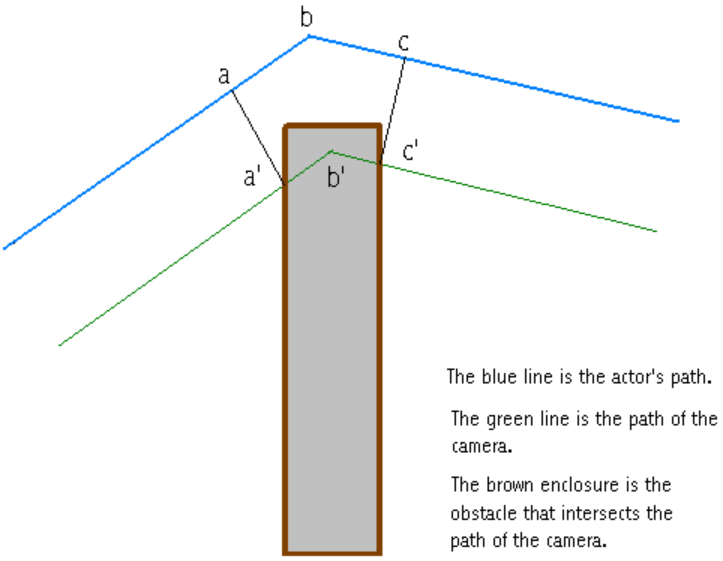


Figure: this shows the condition of operation of the ObstacleIdiom. The path of the camera passes through the obstacle thus leading to occlusion. To avoid this a point of view fragment is used, the camera following the actor along the actor's path for the duration during which the actor moves along the path abc.

References

[1] **Mukerjee, Amitabha, Kshitij Gupta, Siddharth Nautiyal, Mukesh P. Singh and Neelkanth Mishra:** 2000, Conceptual Description of Visual Scenes from Linguistic Models Journal of Image and Vision Computing, Special Issue on Conceptual Descriptions, March 2000 volume 18.

[2] **Salesin, David H., Sean E. Anderson, Li-wei He, David B. Christianson, Daniel S. Weld and Michael F. Cohen:** 1996 Declarative camera control for automatic cinematography. Proceedings of AAAI '96 (Portland, OR), 148-155, 1996.

[3] **A Gift for a Millennium**HIroshi Uchida, Meiying Zhu, Tarcisio Della Senta.

[4] **The Universal Networking Language (UNL) Specifications Version 3 Edition 1**, UNL Center UNDL Foundation.

[5] **Geib, Christopher, Libby Levison and Michael B. Moore:** 1994 Soda Jack: An Architecture for Agents that Search for and Manipulate Objects

[6] **Yamada: 1993,** Studies on Spatial Description Understanding based on Geometric Constraints Satisfaction, PhD thesis, January 1993, Kyoto University.

[7] **Badler, Norman I. and Stephen W. Smoliar**: 1979, Digital Representations of Human Movement, ACM Computing Surveys, March 1979 volume 11.

[8] **Hoffman, Christoph M. and John E. Hopcroft**: 1987, Simulation of Physical Systems from Geometric Models, IEEE J. of Robotics and Automation, June 1987 volume RA-3

[9] **Lozano, Oded Maron Tomas and Tomas Lozano-Perez** : 1996, Visible Decomposition: Real Time Path Planning in Large Planar Environments, AI Memo.