

A VISUALIZATION TOOL TO STUDY THE MOTION
OF COMPLEX 3D OBJECTS IN SPACE

A Senior Honors Thesis

by

BHARATINDER SINGH SANDHU

Submitted to the Office of Honors Programs
& Academic Scholarships
Texas A&M University
in partial fulfillment of the requirements of the

UNIVERSITY UNDERGRADUATE
RESEARCH FELLOWS

April 2003

Group: Engineering & Physics 2

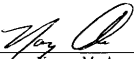
A VISUALIZATION TOOL TO STUDY THE MOTION
OF COMPLEX 3D OBJECTS IN SPACE

A Senior Honors Thesis
by
BHARATINDER SINGH SANDHU

Submitted to the Office of Honors Programs
& Academic Scholarships
Texas A&M University
in partial fulfillment for the designation of

UNIVERSITY UNDERGRADUATE
RESEARCH FELLOW

Approved as to style and content by:



Nancy M. Amato
(Fellows Advisor)



Edward A. Funkhouser
(Executive Director)

April 2003

Group: Engineering & Physics 2

ABSTRACT

A Visualization Tool to Study The Motion
of Complex 3D Objects in Space. (April 2003)

Bharatinder Singh Sandhu
Department of Computer Science
Texas A&M University

Fellows Advisor: Dr. Nancy M. Amato
Department of Computer Science

Visualization is the process of mapping numerical values onto perceptual dimensions. Visualization is a way to show results in a manner that is intuitive to humans. Pictures and animations presented properly can be easier to understand than numbers generated by a computer program. Graphical representation of ideas and results allows a larger audience to understand and appreciate them.

Motion planning consists of moving an object (robot) from one configuration to another. In motion planning we are concerned with the path a robot takes to reach the goal position. Even in non-robotic applications like protein folding, we are concerned about the path the protein takes to fold to its native fold. Visualization allows us to see the robot interacting with its environment in the process of reaching the goal. This enables us to determine the quality of paths. It also allows humans to "tweak" the computer-generated paths to make them better by adding critical nodes that the computer might have missed. For complex robots and environments, visualization tools can help present the program output in a comprehensible format.

The main objective of this project is to create a visualization tool (Vizmo++) to serve as an interface between motion planning algorithms and the researchers who use and design them. Vizmo++ will enable researchers to model their complex robots and environments and to see the results of their motion planning algorithms interactively.

The software will simplify the construction of complex environments and will aid in visualizing the "paths" taken by their robot from one configuration to another.

Vizmo++ is being designed in a highly extendable fashion to allow easy future expansion. The user interface is being designed so that the software is intuitive to use. Vizmo++ now allows researchers to study the solutions to complex motion planning problems by stepping through animations. It also enables them to save the animation for use in movies and presentations. Scientists can now view the entire path the robot takes to go from the start to the goal position. They can also change the representation of the robot to a point or cube if so desired. Vizmo++ is currently being enhanced to support a wider array of robots. In the future it will be possible to assemble motion planning problem environments in Vizmo++. Ultimately, Vizmo++ will serve to create and present motion planning problems along with their solutions in an easy to understand fashion.

ACKNOWLEDGMENTS

I would like to thank Dr. Nancy M. Amato, my advisor, for guidance. Thank you for your advice and constant encouragement.

I would also like to thank the members of the Parasol Motion Planning group for their help and ideas: Burchan Bayazit, Nick Downing, Jinsuck Kim, Jyh-Ming Lien, Marco Morales, Rick Stover, Guang Song, Aimee Estrada, Kasthuri Srinivasan, Rick Stover, Xinyu Tang, Dawen Xie, Roger Pearce.

TABLE OF CONTENTS

	Page
ABSTRACT	iii
ACKNOWLEDGEMENTS	v
TABLE OF CONTENTS	vi
LIST OF FIGURES	viii
CHAPTER	
I INTRODUCTION	1
A. Motion Planning	1
B. Configuration Space	2
C. Probabilistic Roadmap Methods and Obstacle Based PRM (<i>OBPRM</i>) Package	2
D. Visualization	5
II PREVIOUS WORK	8
A. Vizmo 1.0	8
B. Vizmo 1.0 Drawbacks	8
III VIZMO++	10
A. Framework	10
B. Incorporating Existing Technologies	11
C. Vizmo++ Components	12
D. Vizmo++ Programming Manual	14
1. The Plum Module	14
2. The Source Module	16
3. The GL Module	16
4. The Math Tool Module	17
IV APPLICATION: CAMPUS NAVIGATOR	18
A. Components of the Campus Navigator	18
1. Overview of Campus Navigator	19

CHAPTER	Page
2. The Campus Graph	19
3. The Roadmap Editor	20
4. Campus Graph Query	21
5. Campus Path Visualization Via Vizmo++	21
6. User Web Interface	22
V CONCLUSIONS	24
A. Summary of Work Done	24
B. Future Work	24
REFERENCES	26
VITA	28

LIST OF FIGURES

FIGURE		Page
1	OBPRM	3
2	OBPRM Motion Planning Package Framework	4
3	Sample Model File for a 3D Cube	5
4	Sample Environment File for three Multibodies	5
5	Sample Query File	6
6	Sample Map File	6
7	Sample Path File	7
8	Vizmo 1.0	9
9	Vizmo++ Architecture	12
10	Vizmo++ being used to analyze the solution of a motion planning problem	15
11	Overview of Campus Navigator	19
12	Campus Navigator Prototype Interface using Vizmo++	22
13	Campus Navigator Prototype	23

CHAPTER 1

INTRODUCTION

Motion planning consists of moving an object (robot) from one configuration to another. Recently, automatic motion planning has been applied to many areas such as robotics, virtual reality systems and computer-aided design, and even computational biology.

Several kinds of motion planners exist. Deterministic techniques work for cases that are low dimensional and very simple [1] [2]. However, it is computationally infeasible for deterministic techniques to perform motion planning for many realistic situations [2]. To do motion planning for complex robots in cluttered environments, Probabilistic Roadmap Methods (PRMs) can be employed [3]. The Parasol Motion Planning group at Texas A&M University has developed a number of PRM variants for motion planning, e.g., obstacle-based PRM [4], medial axis PRM [5], closed chain PRM [6], and Customizable PRM [7] to name just a few.

A. Motion Planning

A lot of research has been conducted in the area of automatic motion planning. Motion Planning involves finding a collision free path to move an object from a start configuration to a goal configuration. A configuration of a robot refers to a unique position and orientation.

Since computers lack the intuition necessary to plan paths like humans, various computational techniques have been employed to solve motion-planning problems. A deterministic motion planner is complete, i.e., it guarantees that a solution will be

The journal model is *IEEE Transactions on Automatic Control*.

found if one exists. However they are only effective for simple motion planning problems, i.e, robots with a few degrees of freedom (dof) in uncluttered environments. To solve more complex problems involving a large number of dof with cluttered environments containing narrow passages, various probabilistic algorithms have been employed. While probabilistic algorithms are not complete (i.e., they are not guaranteed to find a solution if one exists,) they are more efficient at finding solutions to harder motion planning problems.

B. Configuration Space

A configuration of a robot refers to a unique position and orientation. The Configuration Space (*C-space*) is a multi-dimensional space where the dimensions represent the *dof* of the robot. Therefore, a configuration contains all information required to describe the robot's position and orientation in the real world. A valid point is a configuration where the object is not colliding with itself or an obstacle. The configuration of a single point in three-dimensional space has three dimensions - its location along the x-axis, y-axis and z-axis. For a cube moving in three-dimensional space, the C-space is 6 dimensional- its location along the x-axis, y-axis, z-axis, roll, pitch and yaw angles. As a result, as the robot becomes more complex, more parameters are needed to define its configuration.

C. Probabilistic Roadmap Methods and Obstacle Based PRM (*OBPRM*) Package

Probabilistic Roadmap Methods (PRMs) [3] can be employed to solve motion planning problems. Random configurations are generated and the collision free configurations are retained. These free configurations are then connected using simple "local" planners to form a roadmap. In the query phase, the start and goal configurations

are connected to the roadmap and the shortest path connecting the start and goal configurations is extracted from the roadmap.

The Motion Planning group at Texas A&M University has developed a number of PRM variants for motion planning, e.g., obstacle-based PRM (*OBPRM*) [4], medial axis PRM [8], closed chain PRM [6], and Customizable PRM [7] to name just a few. *OBPRM* is a variant of PRM where the nodes are generated along the surface on the *C*-obstacles. These correspond to configurations in which robot is in contact with an environment obstacle (see Figure 1) This helps in finding paths in narrow passages which is one of the major challenges in automatic motion planning.

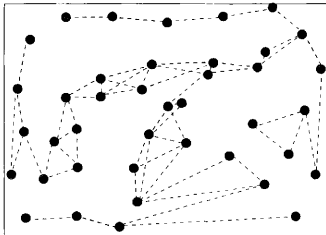


Fig. 1. *OBPRM* generates and connects free configurations along the *C*-obstacle surfaces.

The group utilizes an internally developed package that incorporates all these algorithms. The basic framework of the package consists of three input files, two output files and two programs (see Figure 2.) The input files include:

Model Files (.g)* : These files define the robot and obstacles in the workspace in BYU (Brigham Young University) format. An example is shown in Figure 3.

Environment Files (.env)*: are a collection of model files which together define

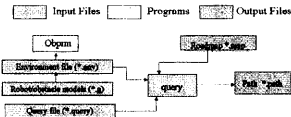


Fig. 2. OBPRM Motion Planning Package Framework

the motion planning problem's environment. An example is shown in Figure 4.

Query Files (.query)*: This file contains the start and goal configuration of the robot. An example is shown in Figure 5.

The model files and the environment file are fed into the *obprm* program and a *Map* file is produced. See Figure 6.

Map Files (.map)*: The map file is produced by the OBPRM program and stores the roadmap for a specific environment. See Figure 7.

The map file and the query file are fed into *query* program and a *Path* file is generated.

Path Files (.path)*: The path file contains the path, a set of configurations linking the start and goal configurations. See Figure 8.

As we can see from the figures, the formats of the files used are very hard to understand. Therefore we need a way to visually present the numbers in these files to human users.

Fig. 3. Sample Model File for a 3D Cube

1	8	12	36
1	12		
1	1	1	
1	1	-1	
1	-1	1	
-1	1	1	
-1	-1	-1	
1	5	7	
-	-	-	-
-	-	-	-

Fig. 4. Sample Environment File for three Multibodies

```

Multibody Active
4
FreeBody 0 cube.g 0 0 0 0 0
Freebody 1 link1.g
Freebody 2 link2.g
Freebody 3 link3.g
Connection
3
0 1 Actuated
- - -
- - -

```

D. Visualization

Visualization is the process of mapping numerical values onto perceptual dimensions [9]. Visualization is a way to show results in a manner that is intuitive to humans. Pictures and animations presented properly are easier to understand than numbers generated by a computer program. Graphical representation of ideas and results allows a larger audience to understand and appreciate them.

In motion planning we are concerned with the path a robot takes to reach the goal position. Even in non-robotic applications like protein folding, we are concerned

Fig. 5. Sample Query File

```
0 0 0 0 0
20 5 10 0.2 0.8 0.5
```

Fig. 6. Sample Map File

```
Roadmap Version Number 061300
#####PREAMBLESTART#####
./obprm -f narrow -cd RAPID -gNodes ....
#####PREAMBLESTOP#####
#####ENVFILESTART#####
narrow.env
#####ENVFILESTOP#####
#####LPSTART#####
3
--
--
```

about the path the protein takes to fold to its native fold [10]. Visualization allows us to see the robot interacting with its environment in the process of reaching the goal. This enables us to determine the quality of paths. It also allows humans to "tweak" the computer-generated paths to make them better by adding critical nodes that the computer might have missed. In the case of complex robots and/or environments it becomes necessary for some kind of visualization tool to present the program output in a comprehensible format.

Fig. 7. Sample Path File

```
VIZMO.PATH.FILE Path Version 20001125
1
763
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
-0.078307 0.002790 -0.060100 0.000005 0.000038 0.001176
-0.156614 0.005581 -0.120200 0.000009 0.000076 0.002352
-0.234922 0.008371 -0.180300 0.000014 0.000114 0.003527
-----
-----
```

CHAPTER II

PREVIOUS WORK

A. Vizmo 1.0

The Parasol Motion Planning group has utilized various visualization tools for motion planning analysis. Visualization of robot paths was first accomplished by using Product Vision (PV) [11], developed by GE at their Corporate Research and Development Center. PV was used primarily for path animation of the robots. However, creation and display of paths using PV was cumbersome. It was also slow and user response and animation were not good [2]. Finally, it only applied to 3D rigid objects.

Vizmo 1.0 (See *Figure 8*) was created by Renu Isaac, Department of Computer Science, Texas A&M University for an MCS project, under the guidance of Dr. Nancy Amato. It is a display tool for paths representing a robot's motion in an environment. It allows designers to create and manipulate environments in 3D space. It also enables the users to generate motion-planning queries and display the results on the screen.

B. Vizmo 1.0 Drawbacks

VIZMO 1.0 is still used in Motion Planning group. However its effectiveness in visualization for the current research projects in the lab has been severely hampered due to its old design and constraints. The design of the software is not suitable for easy extension. As a result it has become increasingly difficult to support new research topics robots such as deformable robots, articulated robots, proteins, neuron and cortical network models. It cannot be used to view configuration spaces and it is not easy to add support for haptic devices (virtual reality devices which provide users with a sense of touch). VIZMO 1.0's interface is also old and not very intuitive

to use. See Figure 8.

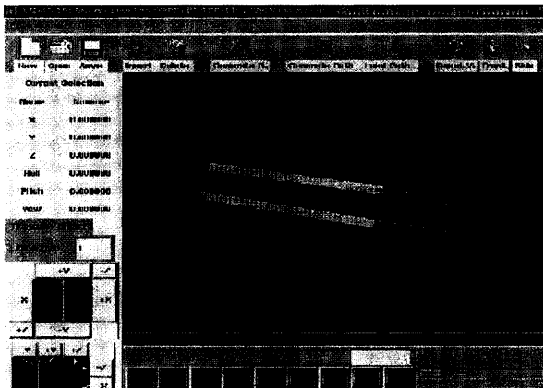


Fig. 8. Vizmo 1.0

CHAPTER III

VIZMO++

Vizmo++ has been designed to replace Vizmo 1.0 for use in animating motion planning problems. It has been designed in a manner to overcome the shortcomings of Vizmo 1.0. Its design also allows for easy extension for future needs. Vizmo++ will enable researchers to view their complex robots and environments using this tool and to see the results of their motion planning algorithms interactively. The software will simplify the construction of complex environments and will aid in the visualization of the paths taken by their robot to go from one configuration to another. Its intuitive interface is easy to understand and use.

A. Framework

Vizmo++ has been designed broadly with the two main component types `ILoadable` and `GLModel` (see *Figure 9*.) The `ILoadable` module loads the different file formats into internal data-structures and the `GLModel` module contains the data-structures for visual representation of the objects read in from the files. The environment, object and map loader modules inherit the `ILoadable` module. The environment loader module reads in the environment file, the map loader module loads the map module, and object loader module reads in the object files into internal data structures. The bounding box and the path computed from query also inherit the `ILoadable` abstract class.

To display the information in 3D format the Environment Model object, Map Model object, Bounding Box Model object and Path Model object all inherit from the `GLModel` abstract class. These object types contain functions to map the BYU format into its corresponding 3D representation on screen. The Plum Object (named

after the fruit) is the all encompassing module and the functions in different modules are invoked through the Plum object. Therefore the plum object acts like an interface for all the underlying modules. The GUI and events are managed by the Vizmo++ module which uses Plum to display the 3D information.

This design is highly extensible. Support for new robot and environment types can be easily added by creating new classes that extend ILoadable and GLModel. Also, all the functionality is divided between different modules and this makes it easy to modify modules independently of the others. The Plum Object acts like a black box and only permits valid function calls. This allows the developers to modify and extend the GUI and event handling without affecting the underlying architecture.

B. Incorporating Existing Technologies

Vizmo++ utilizes existing and proven technologies to accelerate development time, improve quality and reduce testing time. OpenGL (www.opengl.org) was used as the 3D engine and Qt (www.trolltech.com) was employed for developing the GUI.

OpenGL has been used as the 3D rendering engine. OpenGL is a standard and has been employed in developing scientific and computer game software. It is free and is well supported over different platforms. It is also easy to use and its use facilitated quick mapping of numbers from the files to 3D shapes on screen. It also has support for mouse events that were incorporated into Vizmo++ to make it interactive. The use of OpenGL has helped Vizmo++ to create quality graphics at a fast pace. Finally, Vizmo++ will be able to take advantage of future extensions of OpenGL and increase its functionality and power.

Qt, developed by Trolltech www.trolltech.com, is a powerful GUI library. It has many prebuilt GUI components which are easy to add. It signals and slots mechanism

has been useful for increasing the interactivity of Vizmo++. The use of Qt for GUI development has enhanced the Vizmo++ interface and has made it intuitive to use.

C. Vizmo++ Components

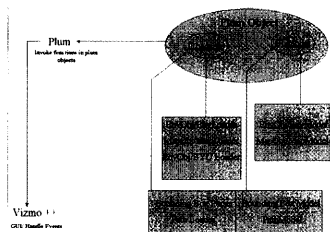


Fig. 9. Vizmo++ Architecture

The motion-planning problem is usually displayed in a 3D fashion projected onto a 2D screen. The user can interact with the environment and its objects by clicking on them. The mouse can be used to zoom in and out, to rotate the environment and to select objects. This helps in a closer evaluation of the solutions found by the motion planning algorithms. There is also support for bubble help (small help windows that pop up when the user hovers over a button) to assist new users by displaying the function of the buttons when they hover on top of them.

The user opens the environment through the file menu. When an environment file is opened the corresponding roadmap and path files are opened too. The roadmap and path can be displayed by either clicking on their icon in the icon tray or through the menus. For perspective, an axis is displayed on the bottom right corner. The

colors of the objects can be changed along with their rendering. They can be rendered as solid, wire frame or hidden. In this way, the bounding box can also be hidden if desired. The objects and background color can also be changed. This can help produce quality images for publications, presentations and demonstrations.

Vizmo++ also includes a tree view control system which presents a hierarchical view of the objects. All the objects comprising the environment can be seen and selected by clicking on the environment link. The individual objects can then be selected by clicking on them. After an object is selected, its information is displayed onto the screen and its appearance can then be altered. This mechanism provides an easy and fast way to view and select various elements of the motion planning problem and its solution.

To animate the solution found by the motion planning package, the path the robot takes to go from the start configuration to the goal configuration can be played back onscreen. Vizmo++ contains a VCR control panel that can be used for playback purposes. It displays the total number of configurations in the path and the step size (the number of configurations to skip during playback to control the speed of animation). Users can also step through the configurations one at a time in any direction. The motion of the robot can be played back in the two directions forward and backward. There are two mechanisms to jump to a particular step: by entering the step number in the Frame field or by moving the slider bar to the particular step. Again, the slider bar can be used to closely examine the robot interacting with its environment by moving it back and forth.

Vizmo++ also provides support for saving screenshots and movies of the motion planning problem and its solution. It allows the user to either save the entire 3D space or select a region to space. The user can save an image of a particular region by drawing a rectangular box and then taking a screenshot. Also, the animation of

the solution can be saved to a file. Vizmo++ allows the user to specify the start step, end step and the step size for the movie. While the movie is being saved, a progress bar is displayed on the screen. Vizmo++ uses the convert application to convert images from ppm format to various other formats like jpeg, gif, bitmap, etc. supported by the convert application. Therefore, in the future convert can add support for additional picture formats.

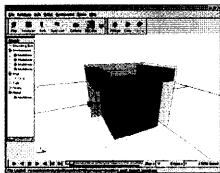
The toolbars and panel can be moved around, docked and customized by the user. With the use of standard and established technologies like OpenGL and Qt, Vizmo++ is portable and in particular can be run on windows and linux operating systems. Its modular and extensible design allows it to be easily expandable for future requirements. As a result of the improvements Vizmo++ is now used in the Motion Planning group lab for visualization purposes. Figure 10 shows Vizmo++ being used to study the solution of a motion planning problem.

D. Vizmo++ Programming Manual

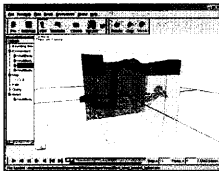
Vizmo++ contains the following main modules: *Plum*, *Source*, *GL*, and *Math Tool*. This section will describe these modules and show their inter-relation with each other.

1. The Plum Module

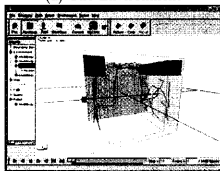
The *Plum* module forms the guts of Vizmo++. It contains the *GLModel*, *ILoadable*, *Environment*, *Map*, *Plum Utility* and *Plum State* objects. *ILoadable* and *GLModel* are abstract classes inherited by *Environment* and *Map* objects. The *Plum utility* object initiates the *Environment* and *Map* objects. The *Environment* object consists of *Environment Loader* and *Environment Model* objects. The *Environment Loader* object inherits from *ILoadable* and parses the environment file. The *Environment*



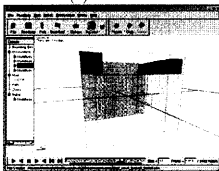
(a) The Environment



(b) The Path



(c) The Roadmap



(d) Animation of the solution

Fig. 10. Vizmo++ being used to analyze the Solution of a Motion Planning problem

Model object inherits from GLModel and contains data structures for visual representation for the motion planning environment objects. Similarly, the Map object consists of the Map Loader and Map Model objects that inherit from ILoadable and GLModel respectively. The Plum State object contains the state information for the Plum object. It contains object rendering information (i.e., if the objects are displayed in solid, wireframe or invisible mode) and their display color. It also contains information about any errors that might have occurred during the parsing of files or rendering of the objects. Overall the Plum module encapsulates the ILoadable and GLModel parts of the Environment and Map objects and facilitates communication

between them.

This design is easy to extend. When the files are parsed, a configuration object corresponding to the motion planning problem is initiated. To add support for new types of environments, only new configuration classes need to be created and integrated into the Environment and Map objects. As a result, different motion planning problem types remain isolated from each other in code because of different configuration classes. When parsing the environment and map file, the appropriate configuration class is automatically initiated. This design has resulted in a clean and a highly extendable code.

2. The Source Module

The *Source* module includes the the *main* function and contains code to render the robot and the bounding box. It also contains the GUI object. The GUI object consists of various GUI element classes such as VCR control, snapshot toolbar and tree view control. Each of the elements have their own classes and are initiated in the main GUI class. Therefore, additional GUI elements can be added and existing GUI elements can be modified independent of each other making the code extensible. The Qt libraries are used extensively in this module.

3. The GL Module

The *GL* module sets up the global 3D rendering environment by creating camera, lights and the perspective axis. It also initializes the picking box which can be used to select multiple objects. When a file is loaded the camera is focused so that the bounding box appears centered on the screen. Lastly it enables mouse and keyboard events. This module makes extensive use of the OpenGL libraries.

4. The Math Tool Module

The *Math Tool* module contains the code for performing various mathematical computations. They include matrix operations, Euler Angle and other vector calculations necessary for 3D visualization.

These modules together make up Vizmo++. Makefiles are provided that produce the executables.

CHAPTER IV

APPLICATION: CAMPUS NAVIGATOR

There are a number of applications of motion planning research. One application currently under development in the Parasol Lab is a campus path planner. This program will allow users to find their way across the Texas A&M University campus. Essentially, the campus navigator is similar to applications such as Yahoo! Map and MapQuest in that it provides users with directions (and a image of the route) to get from one location on campus to another.

However, the campus navigator goes beyond the simple point-to-point route planning of these existing map programs. The campus navigator is designed to allow much more sophisticated queries tailored to the specific needs of the user. For instance, the campus navigator takes transportation mode changes into consideration. The user can specify if she will be walking, riding a bike, driving a car, or willing to take the bus.

As an example, consider a user wishing to find a route to get from a building on main campus to a building on west campus. There are a number of ways to accomplish this. One could simply walk to west campus. Using the campus navigator system, the user can find which bus(es) to take, where and when they pick up, saving time and effort. The system will take into account driving conditions (i.e. closed streets due to construction), parking lots based on permit restrictions, and handicapped accessibility to provide the best path for the user.

A. Components of the Campus Navigator

Currently, the campus navigator is under development. A prototype of the system is expected to be ready by the end of the Spring 2003 semester. The next sections

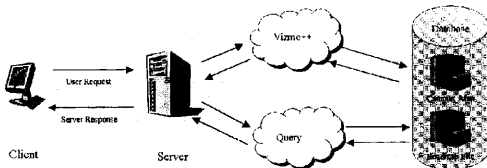


Fig. 11. Overview of Campus Navigator

describe the four fundamental components of the campus navigator system.

1. Overview of Campus Navigator

Before delving into each of these components, an overview of the system as a whole is in order. The user will interact with the system through a set of web pages. These pages allow the user to specify the start location and destination. The user's selection is sent to a program, query, that searches a preconstructed graph of campus (created via the roadmap editor). This graph, which is stored in a database, contains all the data needed by query to select a route that meets the user's request [12].

The path resulting from query's search of the graph is sent to Vizmo++. Using a 3D model of campus and the path, Vizmo++ creates a JPEG image that depicts the route through campus. This JPEG image is sent back to the user's browser. In addition to the image, a textual description of the route is provided.

2. The Campus Graph

The fundamental component of the campus navigator is a graph. As with many motion planning problems, this application is built upon the idea of finding a path

through a graph. All the various constraints are expressed through properties of the vertices and weights on the edges of this graph. Vertices are used to represent physical places on campus such as buildings and parking lots. The edges of the graph are used to represent streets and walking paths through campus.

For this application, the graph is stored in a database. The current implementation employs the open source MySQL database management system. There are a number of reasons for storing the graph in a database. First, it allows concurrent access to the graph from the various components of the system. Initial designs called for the graph to be stored in a file, which is a customary storage medium for graphs.

Second, the database simplifies the sharing of data between the campus navigator system components. As an example, the roadmap editor, query, and the web interface need to access building names. The roadmap editor uses the names to allow vertices to be associated with buildings. Query employs building names when generating textual directions, and the web interface needs the names to give the user a list of buildings to choose from (for specifying start and/or destination). Each of these components are currently implemented in disparate languages. The database provided the simplest medium through which all three pieces could access the same data.

3. The Roadmap Editor

Currently, the campus graph is constructed manually. This is somewhat ironic as much of the work in motion planning is aimed at automatically creating a roadmap. Automatic construction of the campus graph is not realistic as most autogeneration techniques rely on randomly generating vertices and connecting those vertices with edges. Random placement of vertices is not appropriate for this application, as vertices need to be tied to specific points on campus. For instance, a vertex needs to be associated with each building and parking lot.

To ease the construction of this large graph, a roadmap editor is currently under development. This program allows the campus graph to be built over an image of the campus. The user of this program can place vertices at each of the buildings, parking lots, intersections, etc. on campus by simply clicking on the building, parking lot, or intersection in the image. Then edges can be added for the streets and walking paths between these vertices. All of this information is stored in the centralized database.

Even if the campus graph could be generated randomly, all the properties of the vertices and edges of the graph would have to be manually specified. For instance, for a vertex representing a parking lot, someone must specify which permits (student, faculty/staff, etc.) are allowed to park in the lot. This is the second role of the roadmap editor. It allows this information to be entered for all the parts of the graph.

4. Campus Graph Query

The query program is responsible for finding routes through the graph that meet the user's request. The system uses Markov-like states and flexible goal states so that general optimization criteria can be used [12] [13] [14]. This application employs a modified Dijkstra's algorithm, which enables one to consider more general optimization criteria and relaxed definitions of the goal state, to find the optimal path contained in the roadmap through campus [13] [14]. Running as a service, the web interface will send requests to query and receive the computed path for Vizmo++ to display.

5. Campus Path Visualization Via Vizmo++

Vizmo++ is responsible for generating a picture of campus with the path overlaid. After the user selects the start and goal points and query generates a path file,

Vizmo++ opens up a 3D model of the campus and the path file to begin creating the snapshot. The camera then zooms onto the path and creates the image. This image is sent back to the web server to be displayed on the user's browser.

6. User Web Interface

Users of the campus navigator will interface with it via a set of web pages. Users will select the start and destination via selection boxes populated by data from the database. From the user selections, the web pages determine the vertices that correspond to the selected locations. These so-called start and goal vertices are given to query which finds the route. The web pages receive an image of the route from an image generated by Vizmo++.

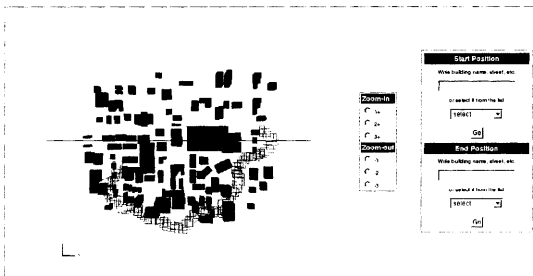


Fig. 12. Campus Navigator Prototype Interface using Vizmo++

Initially, the web interface will provide simply a two dimensional map of campus with the path overlaid. Similar to MapQuest and other programs, the campus navigator web interface allows users to zoom in and out on the returned campus

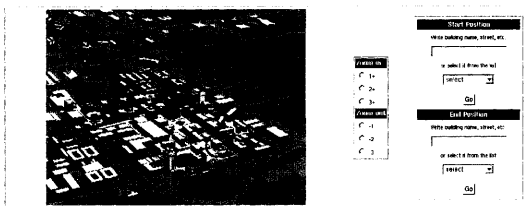


Fig. 13. Campus Navigator Prototype Interface

path. Ultimately, it is envisioned that users would be able to generate a movie, allowing the user to "fly-through" a 3D model of campus along the path generated by query.

The campus navigator is an interesting and useful application of motion planning research. Exploiting the techniques developed to plan the paths of robots, the campus navigator aims to guide people around the large campus of Texas A&M University. It is envisioned that this application would potentially be useful for cities. By taking into consideration all the various modes of transportation such as buses and subways, the campus navigator could be extended to a city navigator, allowing residents and visitors to efficiently navigate the city.

CHAPTER V

CONCLUSIONS

Vizmo++ has been developed to animate the solutions found by the motion planning algorithms developed at the Motion Planning group at Texas A&M University. The Campus Navigator is being developed as an application that makes use of the research done with Vizmo++ and the motion planning algorithms.

A. Summary of Work Done

The research focused on developing a 3D Visualization tool to study motion planning problems. The solution developed, Vizmo++, allows researchers to model robots and environments and to see the results of their algorithms interactively. Vizmo++ has been designed in a highly extensible fashion which will allow easy future extension. The user interface is also very intuitive to use.

Campus Navigator is a tool to find directions for users on Texas A&M Campus. It allows the users to customize the path by choosing various modes of transportation and by further tailoring the search, e.g., by specifying that they want to avoid dimly lit paths after 5pm.

B. Future Work

Vizmo++ is designed to be used for visualizing all motion planning problems solved in Motion Planning group. To achieve this, it has to support articulated robots. Articulated robots are composed of multiple links joined together. Then Vizmo++ can be used to model and view high dimensional objects.

Currently, Vizmo++ can only used be used to visualize the motion planning

problem and its solution. There is no integration with the OBPRM package. A Future goal would be to allow the user to change the start and goal position. Vizmo++ should then rerun the query program and display the new path file.

Other future work might include a robo cam where the camera is positioned on the robot and users get an option for a view of the animation from the robots perspective. Currently users can change the color of the objects; in the future they should be also allowed to apply textures to the objects. Support for haptic devices will add force feedback to the visualization. This can be used to position the individual nodes through the sense of touch. Also functionality for Visualization of 2D and 3D C-space can be added in the future.

REFERENCES

- [1] Y. K. Hwang and N. Ahuja, "Gross motion planning – a survey," *ACM Computing Surveys*, vol. 24, no. 3, pp. 219-291, 1992.
- [2] J. C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA, 1991.
- [3] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566-580, August 1996.
- [4] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1998, pp. 155-168.
- [5] C. Holleman and L. Kavraki, "A framework for using the workspace medial axis in prm planners," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2000, pp. 1408-1413.
- [6] L. Han and N. M. Amato, "A kinematics-based probabilistic roadmap method for closed chain systems," Tech. Rep. TR 00-003, Department of Computer Science, Texas A&M University, 2000.
- [7] G. Song, S. L. Miller, and N. M. Amato, "Customizing PRM roadmaps at query time," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2001, pp. 1500-1505.
- [8] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1999, pp. 1024-1031.

- [9] Marchak et al., "The psychology of visualization," *IEEE Visualization 93*, pages 351-54, pp. 351-54, 1993.
- [10] G. Song, S.L. Thomas, K.A. Dill, J.M. Scholtz, and N.M. Amato, "A path planning-based study of protein folding with a case study of hairpin formation in protein G and L," in *Proc. Pacific Symposium of Biocomputing (PSB)*, 2003, pp. 240-251.
- [11] Renu Isaac, "A tool to visualize the motion of 3d objects in space," *Department of Computer Science, Texas A&M University*, May 1998.
- [12] Roger Allan Pearce, "Optimal motion planning with constraints for mobile robot navigation." *Senior Honors Thesis, University Undergraduate Fellows program, Texas A&M University*, 2003.
- [13] Nancy M. Amato Jinsuck Kim, Roger A. Pearce, "Feature-based localization using scannable visibility sectors," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2003, To appear.
- [14] Nancy M. Amato Jinsuck Kim, Roger A. Pearce, "Extracting optimal paths from roadmaps for motion planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2003, To appear.

VITA

Bharatinder Singh Sandhu began his undergraduate studies in the Fall of 1999. He is pursuing a Computer Engineering (Computer Science Track) Degree from the Department of Computer Science at Texas A&M University.

He works as an undergraduate researcher with the Dynamic Spatial Modelling for Tomorrow (DSMFT) group. DSMFT is a robotics group that specializes in motion planning algorithms, graphics and robotics.

He is interested in robotics especially in the research done in robotics for graphics and animation.

Permanent address:

515 Sector 16C
Chandigarh, India

The typist for this thesis was Bharatinder Singh Sandhu.