

# 3D-VDBM: 3D Visual Debug Monitor for RoboCup

David M. Chelberg, Matthew Gillen, Qiang Zhou, Arvind Lakshmikummar

Laboratory for Intelligent Real-Time and Secure Systems

School of Electrical Engineering and Computer Science

Ohio University

Email: [chelberg@ohiou.edu](mailto:chelberg@ohiou.edu), [mgillen@rm.ece.ohiou.edu](mailto:mgillen@rm.ece.ohiou.edu), [zhou@rm.ece.ohiou.edu](mailto:zhou@rm.ece.ohiou.edu),  
[kla@rm.ece.ohiou.edu](mailto:kla@rm.ece.ohiou.edu)

## Abstract:

This paper describes an architecture (3D-VDBM) for distributed three-dimensional visualization and debugging of distributed agents. This architecture is demonstrated through application to the problem of debugging a collection of intelligent agents that play soccer in a simulated environment for the RoboCup international competition.

**Keywords:** Visual information systems, RoboCup, Distributed Agents, Human-computer interaction

## 1. INTRODUCTION

RoboCup (The Robot World Cup) is an attempt to promote intelligent robotics research by providing a common task for evaluation of various theories, algorithms, and agent architectures. [2] Currently RoboCup consists of four competition tracks: computer simulation league, small-sized robot league, medium-sized robot league, and legged robot league. The architecture we have developed was originally intended to facilitate our efforts in the simulator league. We are currently adapting it to help us with our efforts in the small-sized robot league as well.

The simulator league is a server/client system, where the server controls the game (models physics, sends players their visual information, receives commands from players such as *dash*). Each player (client) program is connected via a UDP socket (which does not guarantee the arrival or the order of packets). The server provides clients with realistic visual information: the player can only see in the direction his head is turned; he gets more accurate data about close objects than distant objects; some objects cannot be seen at all. The client must then use this imperfect data to build his worldview. A monitor program may connect to the server. The monitor displays a 2D representation of the game to a computer screen. Typically clients are run on several different machines, and their only means of communicating with each other is through a *say* command to the server. The *say* command is limited in the same way human speech during play is, it is only heard by nearby players, and it

can communicate only a limited amount of information. During a competition, once a game starts humans may not interfere. Each client must therefore be autonomous.

The simulator league currently uses a 2D monitor, where the players are represented as colored circles and the ball is a white circle. Besides being visually unappealing, this monitor has many disadvantages. Some of them are:

- It does not provide any information on the future course of actions of the players or their intents.
- It does not display the field as seen by a player on the field.
- There is no way to interactively change the view.

To overcome these disadvantages and to provide a tool to debug algorithms visually, we developed a 3D visualization and debugging tool using the Visualization Toolkit [1].

The 3D Visual Debugging Monitor (3D-VDBM) provides solutions to the above problems. The monitor can be used to see what the client sees (by switching the camera to his view). It can also extract information about the current internal state of the client. This is one of the most important features: given the perfect information from the server, and information about the client's internal state (i.e. what he sees, desires, is planning), it becomes much easier to determine why the client behaved the way it did. The client's worldview will never be perfect, and it must act on incomplete information. Therefore, it is of vital importance when debugging a client to find out which part of the client made the mistake.

Besides being more realistic, this monitor has practical advantages that make it ideal for any distributed dynamic system. The information we have been getting about the players is from the server. This is very limited information. With our 3D monitor, the viewer can choose to select any player on the field and get his view of the world. In addition to this, he can also place a camera at any location on the field and observe the game from there. The viewer can also choose any particular player from the field and get information local to him, like his stamina,

his probable actions and the threats he perceives. The information on the player stamina is available from the server, but the other information local to the player can be obtained only from the clients. The implications of this are far reaching. All the low-level algorithms that are written for the players can now be visually evaluated for their efficiency, and efficacy. This makes the 3D monitor a good debugging tool for distributed and real-time applications.

## 2. VISUALIZATION TOOLKIT

VTK is an object-oriented approach to 3D graphics and visualization. Its 3D graphics model is at a higher-level of abstraction than other common rendering libraries such as OpenGL. The visualization model supports many common visualization algorithms.

VTK is a freely available C++ class library; one can download it from Kitware. One can develop a visualization application using VTK in C++, and it is available for multiple platforms. Another strong point for VTK is that it has built-in support for distributed visualization (splitting the visualization task among several machines, to achieve better performance).

## 3. 3D VISUAL DEBUGGING MONITOR (3D-VDBM) FOR ROBOCUP

When trying to design a team for competition in the simulator league, we found it was difficult to understand

why a particular player behaved in a certain way. Since the only tool usually provided for debugging is a 2D monitor (see Figure 2), it is difficult to get information about each intelligent agent that is playing in the soccer game. Simply put, we lacked a suitable tool to debug the client programs during game play.

We decided to develop better visualization and debugging tools for our RoboCup efforts. The tool we developed is called our 3D Visual Debugging Monitor. The design goal was to develop a general architecture for debugging distributed real-time applications, and to apply this to the RoboCup problem.

The default RoboCup monitor program communicates only with the central RoboCup server, not with any of the client processes. Our architecture includes a new line of communication from the client to an observer that will collect and collate information from the server and clients. This collated data can then be forwarded to multiple 3D-VDBMs in real-time (so as to inspect multiple clients at the same time) and can be simultaneously written to a log file for off line study. The log files contain, in addition to the usual position of all objects, additional data such as stamina, auditory information, and overall strategy. All of this information is important because it represents something that the monitor previously had no way to determine, and it has proven very useful during debugging.

The advantages are numerous. First, a 3D monitor can help the developer to *see* what the client sees. This is particularly important, since the visual information a

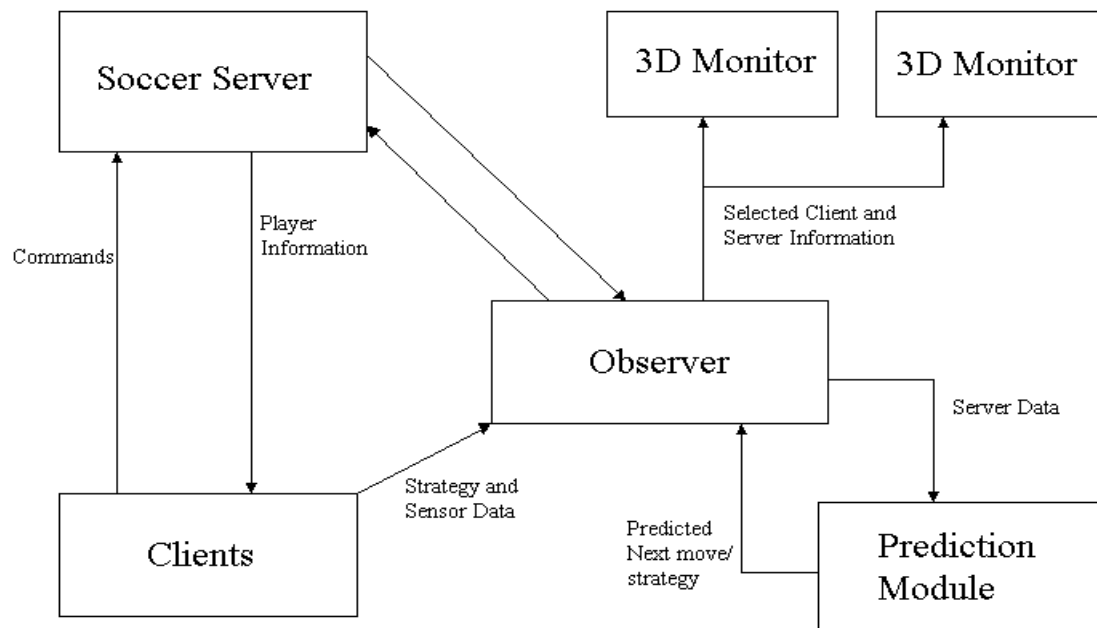


Figure 1: This diagram illustrates the overall architecture

player gets is imperfect, and getting bad information often leads to making poor decisions. Secondly, it makes it easy to select a player and get useful information about his sensors (i.e. what he has heard) and internal state, in addition to what the monitor usually provides (last command executed).

## 3-2.THE OBSERVER

The observer connects to the soccer sever through a normal monitor port. During the debugging/testing phase of our clients, the clients also open a connection to the observer, and send data about the internal state of the agent. The observer can then be used in a distributed environment to forward the information it gathers (from the sever and clients) to multiple 3D-VDBMs running on any machine on the network.

The advantage of using an observer is that only one connection to the server and each of the clients is needed. This not only means that there is a central point for all the data needed by the system, it also helps to keep the design modular (which is important in a distributed environment). It also reduces the overhead of running multiple 3D-VDBMs, since all the input is only formatted once (by the observer).

The information collected by the observer can be useful in other ways. For example, if an agent is behaving in a seemingly abnormal way, one can examine what the internal state of the agent was (as recorded by the observer) at the time of the abnormality. By comparing this with the perfect information from the server (what the agent's world view *should* have looked like), the cause of the anomaly can be narrowed down quickly. .

## 3-3. THE PREDICTION MODULE

Our 3D monitor can be used without special clients designed to communicate with it, but it obviously will not be as useful. We use the 3D monitor to debug our own clients, and when playing other clients (that were not designed for use with the 3D monitor) we would like to have the same information available about the other clients. To solve this problem, we plan on implementing a prediction facility that will track opponents and do things such as evaluate an approximation of what their stamina should be. This is not part of the general architecture, but it is a necessary component for this particular domain.

## 3-4.IMPLEMENTATION OVERVIEW

Our program is divided into two parts, one is an observer and the other is a 3D visualization tool that monitors the overall play and allows users to interactively display the current status of agents that are playing.

We developed the 3D monitor using VTK. The visualization process is interactive: you can zoom in/out and change the point of view to any angle you wish. In VTK there is a special class handling the interactive processing that has the features mentioned above. Each player's view is calculated based on the position of that player. We put a camera where the player is so that we can get the view of that player.

Each player gets visual information every 150ms, and has a cone of vision that spans 90 degrees (by default). The player may change both the frequency of the data and the viewable cone during the game (and there is a record of these commands in the log file). We can use the log file to accurately portray what the player sees.

The view of every player has its own clipping planes, that means objects too near or too far from the player can not be seen, this corresponds with the actual data the intelligent agent gets from the soccer server Our server incorporates a 3D humanoid model for players [10]. This model was originally in 3DS format, which we converted for our use.

The observer gathers information about every player's location, team, number and angle of its head from the server; it also gathers sensor information and information about clients' decisions, and plans from the client. The observer places the information into a structure and passes it to the 3D monitor.

Our architecture calls for a separate observer client and a visualization client -- the 3D monitor. There are several advantages in doing so: first we can have several simultaneous monitors running. Each one could be on a separate machine. This makes it easy to do group analysis of the algorithm. Second, 3D visualization normally will take a lot of computational resources, so separating it from the observer allows the 3D monitors to run on different machines, while keeping an observer running real-time. The writing of a log file is done in the observer module.

The information passed from the client to the observer includes:

- Stamina of the player
- Auditory data
- Contents of *say* commands
- Mode of the client (whether he is in offense or defense)
- Strategy information: high level such as *pass* the ball to player #3 and low level such as *dash* 100% (full power).

The monitor interprets the strategy data into a visual representation and displays it along with the other information about a player coming from the observer.

Figures 4 and 5 illustrate this. In the future we are planning to do the following: We will add a prediction model to generate the extra data that our monitor can use for clients that were not designed to work with the monitor. We will add more lighting effects in the 3D monitor to make the visualization more vivid. We will continue to develop our model for players and design it as a group of objects so that the 3D monitor can perform low-level animation for individual players (such as the *turn\_neck* command).

## 4. CONCLUSION

This paper introduces the 3D-VDBM, a visual debugging tool designed for the RoboCup simulation league. With this tool, debugging of high-level AI algorithms becomes much easier. This architecture can be applied to any complex domain where the massive amount of data makes results hard to interpret. We plan on extending this architecture to help us model our robots for the small-sized league (of RoboCup).

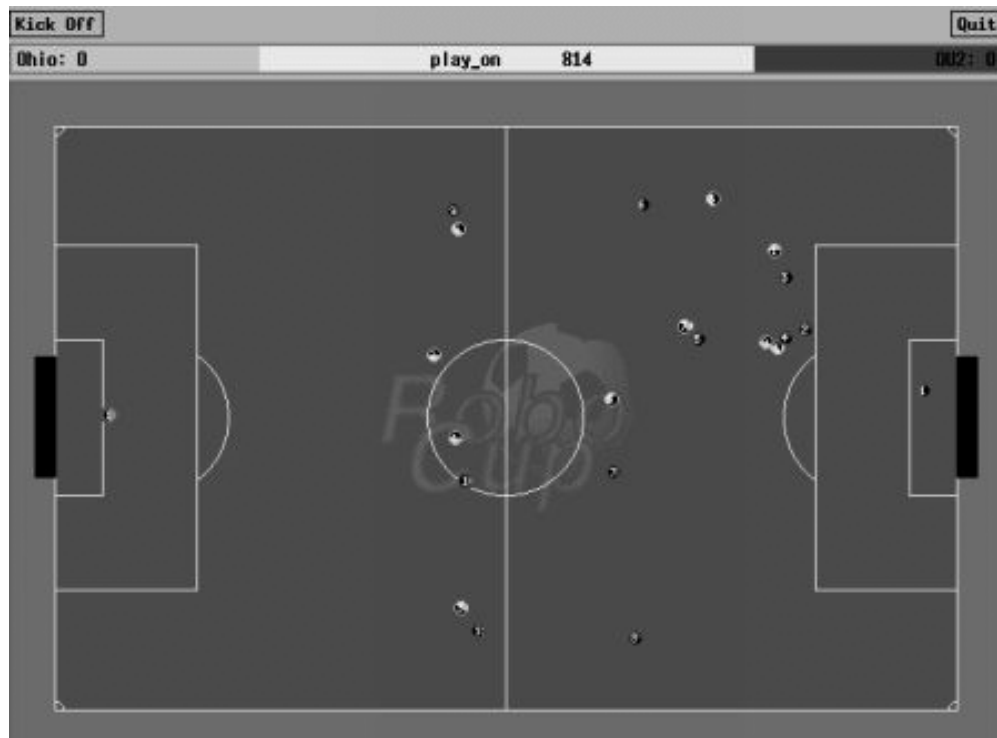
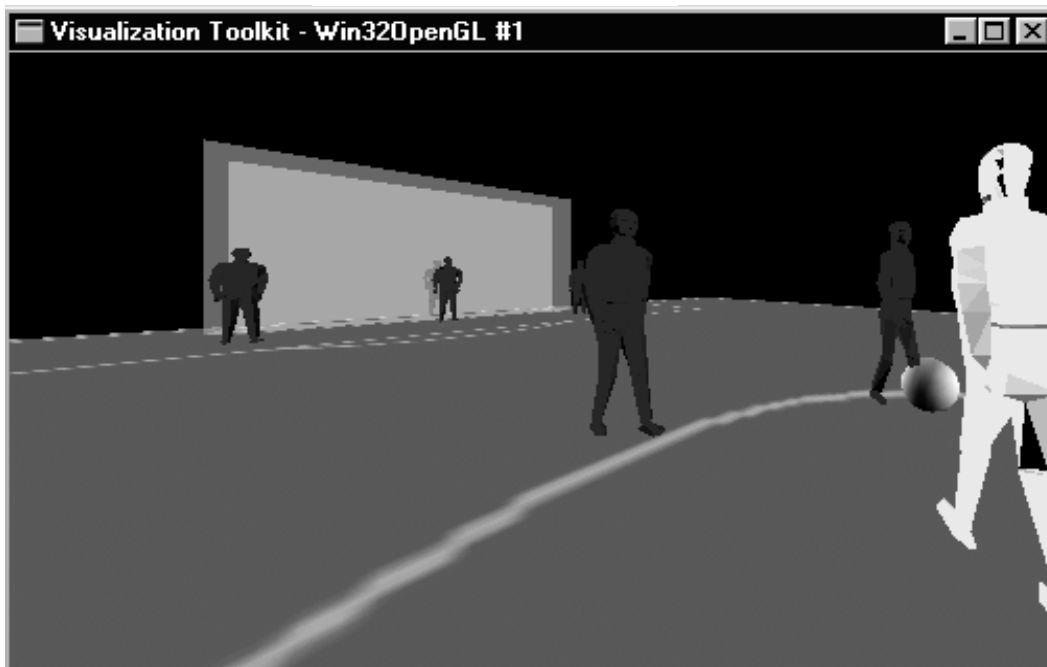


Figure 2: The 2D monitor



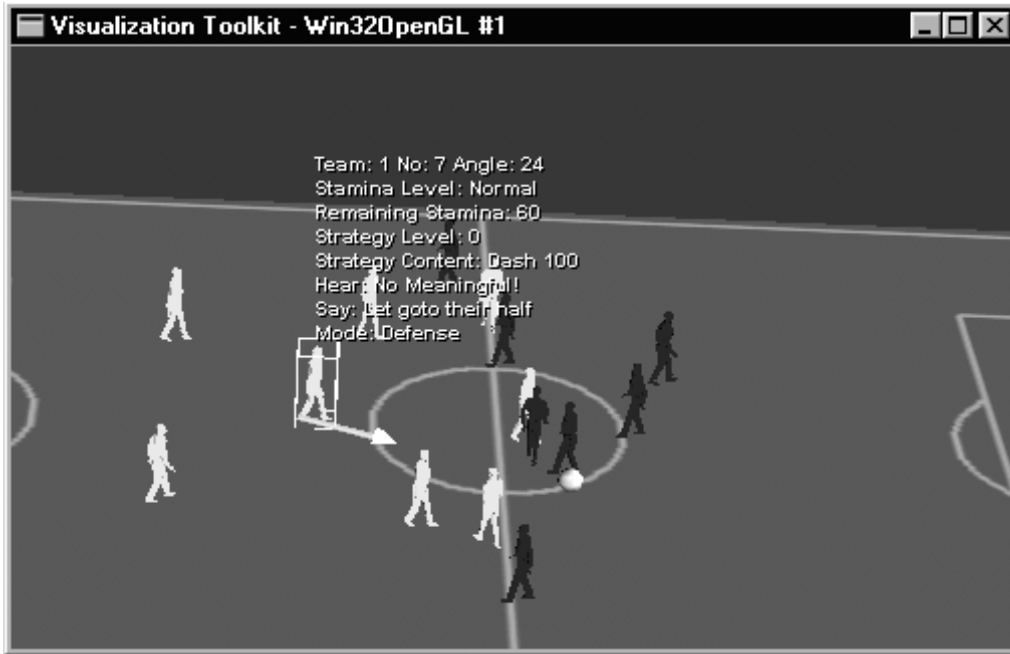


Figure 4: Visual debugging of a distributed real-time agent



Figure 5: Visual Representation of the agent's planned path

## References:

- [1] Ken Martin, Will Schroeder, *The Visualization Toolkit An Object-Oriented Approach to 3D Graphics* Prentice Hall, December 1997.
- [2] Minoru Asada, Hiroaki Kitano, *The RoboCup Challenge Robotics and Autonomous Systems*, Vol 29, p3-12, 1999.
- [3] RoboCup Resource Webpage,  
<http://www.robocup.org/resource/6.html>
- [4] Robocup Server Manual ver 5.0
- [5] Klaus Dorer, Soccer Monitor and Logplayer for Windows, <http://www.iig.uni-freiburg.de/cognition/members/klaus/>
- [6] Bernhard Jung, Markus Oesker and Heiko Hecht, *Virtual RoboCup: Real-Time 3D Visualization of 2D Soccer Games* <http://www.TechFak.Uni-Bielefeld.DE/techfak/ags/wbski/3Drobocup/>
- [7] Patrick Riley, Peter Stone and Manuela Veloso, *Layered Disclosure: Revealing Agents Internals The Seventh International Workshop on Agent Theories, Architectures, and Languages (ATAL-2000)*.
- [8] Peter Stone, Patrick Riley and Manuela Veloso, *Defining and Using Ideal Teammate and Opponent Agent Models Proceedings of the Twelfth Innovative Applications of AI Conference, 2000*
- [9] Peter Stone, Patrick Riley and Manuela Veloso, *The CMUnited-99 Champion Simulator Team RoboCup-99: Robot Soccer World Cup III*, Springer Verlag, 2000
- [10]  
<http://modernmedium.com/free/models/3/inches/3ds3/02/02zman01.zip>