

Automated Commentaries for Simulated Soccer

Justin Hogg, Adrien Martel, Ahsan Mussa, Akbar Sherwani
Department of Computer Science, University of Kent, Canterbury
jh71@kent.ac.uk, am203@kent.ac.uk, am202@kent.ac.uk, as89@kent.ac.uk

Abstract

In recent years there has been a growing interest in robot football with a number of annual tournaments being held and an international standard set of rules being devised. These set of rules have largely been created by RoboCup, an international joint project to promote AI, robotics, and related fields. The overall goal of RoboCup is:

"By 2050, develop a team of fully autonomous humanoid robots that can win against the human world champion team in soccer."

The aim of our project was to automatically generate commentaries for RoboCup games, and play these alongside the game in real time using speech synthesis. This paper will discuss our design and implementation of our automated commentary system. It will begin with a short introduction to robot football before detailing our main aims and objectives, decisions made during the development cycle and how these were implemented before drawing together conclusions about how well these aims and objectives were executed.

1. Introduction

The project was chosen because of our keen interest in football and the range of skills and knowledge each person in the group possessed. As well as good understanding of the Java programming language (the language used throughout this project), we all enjoy and follow football making this an attractive and potentially challenging assignment. The project consisted of:

- Reviewing the small amount of existing work in this area
- Developing a program
- Evaluating the success of the program

One of the main project deliverables was to review the small amount of existing work in the area which we carried out through comprehensive research. We started by finding existing systems that related to our project. To our advantage we found a research paper on three RoboCup simulation league commentator systems – Rocco, Bryne and Mike. They gave us a flavour of what we needed to base our project on, allowing us to use some of the existing ideas as well as using them for post-evaluation. Our approach to developing the project was to use extreme programming (XP). This approach was taken because our

project had never been created before at Kent where we were unable to gather requirements on how and what we wanted the system to contain. XP enabled us to confidently respond to changing system requirements, even at the latter stages. This methodology improved the quality of work produced where we all had the ability to deliver an excellent system. The success of this project was down to four essential factors; communication, simplicity, feedback, and courage.

The evaluation on the success of our project is detailed in the Evaluation section and an overview can be found at the end of this report. The development of our system is broken down into four key functions - Analysing, Speech and Atmosphere, Prioritisation and Visualisation. The functions are broken down separately through this report.

2. Background

2.1 What is RoboCup

The idea of RoboCup is to provide a standard problem for researchers to explore artificial intelligence and robotics where a wide range of technologies can be examined and integrated. The concept of a robotic world cup was first introduced in 1993 and after a two-year feasibility test an announcement was made on the introduction of the first international conferences and football games. In July 1997 the first official conference and games were held in Nagoya, Japan and have been held every year since at different venues around the world and covered by the international media.

There are five main leagues in RoboCup; a simulation league where teams compete on a simulated playing field on computers, a small-sized robot league, a middle-sized robot league, a four-legged robot league and a humanoid league. Due to time and resource constraints the simulation league was the only option available to us. However our commentary system is dynamic and should be able to commentate in various different simulation leagues.

2.2 Soccer Server

Games of football are played on computers using the RoboCup simulator called the soccer server. The RoboCup soccer monitor (see Figure 1) allows visualisations of the game to be displayed on the users screen using the Windows or Linux operating system. The server is written to allow multiple virtual soccer players to connect in an unspecified computing

environment with real time demands as well as semi-structured conditions. This allows researcher to focus their efforts on cooperation and learning techniques rather than the details of connecting.



Figure 1 - The soccer monitor screen

3. Aims

The aim of our project was to automatically generate commentaries for RoboCup games, and play these alongside the game in real-time using speech synthesis.

4. Technical Implementation

The technical implementation of our project will discuss the technical details of each of our main functions that make up our system.

4.1 Analysing

In order to produce commentary our software must determine what is actually occurring on the field. The first challenge was actually to connect to the RoboCup soccer simulation server and retrieve data which was valuable for our system.

Our software was able to connect to the server and use the monitor protocol. The monitor protocol allowed more than one monitor to be connected hence we could use our system while watching the simulation on a 2D visualization monitor.

The monitor protocol meant our software would be sent the following useful data packets every 100ms over a UDP connection:

- Player Details
- Game Details

Player Details:

The player details included each player's positions, player numbers and whether the player was kicking the ball.

Game Details:

This included the team's names, time and the current score line as well as the game state. This also included the ball positions and velocity on the field.

What became apparent before coding began was that the system was going to receive a lot of time critical data as the server sends out packets every 100ms. Therefore from early on in the project we followed principles of a real-time system by continually keeping performance issues and efficiency in mind.

4.1.1 Game Analyser:

After reading through research on other systems it became clear to have a class which would handle the analysing of the data. The aim of the Game Analyser class was to identify set rules and patterns to build up a picture of the game and hence be able to commentate on it.

Example:

Using the game details packet we can identify the game state of the match. For instance if the game state changes to "Goal Kick Left" we can assume the ball was shot by the team on the right and has gone wide of the goal. Using the player details packet we can also identify who was the last player to touch the ball, for this reason we can determine the following:

- Player X shot the ball
- Ball out of play for a goal kick
- Shot off target for team on the right

Using the technique described above we could identify many sets of plays (10 in total) and therefore have a basic level of commentary based on facts and rules of football. However this system was basic and we needed to make the commentary more accurate and rely more on positions of the ball.

4.1.2 Ball Class:

As mentioned earlier, the ball position was passed to us by the server and we used this in our system in the following way.

The ball class kept the detail of the balls' position at all times. Once a player touched the ball, the position of the ball was noted along with the player's number. This information was then added to a hash map such as the example given in Table 1 below.

X: 32 Y: 45	Player 12
X: 45 Y: 13	Player 15

Table 1 – Example of the Hash map

Once a pass was identified the last two touches are taken from the hash map and the distance is computed using the last values above:

1. Confirm that both players on the same side
2. Work out the vector difference between the ball positions and Pythagoras:

Player 12 X – Player 15 Y = X across difference -13

Player 12 Y – Player 15 Y = Y across - 32

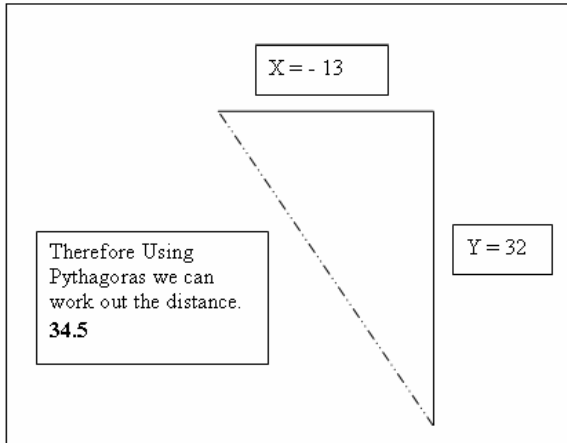


Figure 2 - Pythagoras method for obtaining distance

Hence using Figure 2 above we used the same method to shots on goal by either identifying when the goal keeper touched the ball.

Or if the goal had been scored then we can take the last touch to be the centre of the goal and work out the distance to the goal. This produced commentary whereby we could determine whether long passes were being played and whether shots were being taken from long distances.

Furthermore using the Game Analyser example of the goal kick we could now determine the final ball position when the play mode changed to ‘goal kick’ and derive from this the distance of the ball from the goal posts. Therefore the commentary would react to that extra parameter.

This increased the commentary complexity, nevertheless our commentary only reported on events that had occurred. To improve this, we decided to introduce predictability in some of the future events that may occur.

4.1.3 Voronoi:

Reading through papers based on MIKE, we came across the Voronoi diagram and decided to use a similar version in our system. This diagram analyses the positions of the players and works out the plays and whether the team has a good chance to score in the next few time iterations. Rather than analysing the match and events which have occurred, this is looking for events which may happen in the future, as human commentators would do.

The Voronoi uses the positions of players along with the type of players such as strikers or midfielders and it attempts to predict what is about to occur on the field.

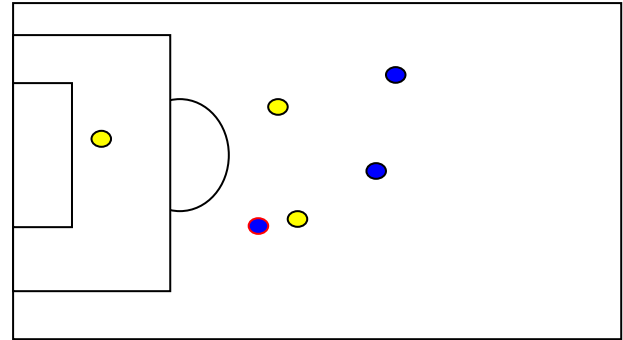


Figure 3 - Voronoi representing 'Beaten last defender/through on goal'

In Figure 3 above, the blue team have possession with the red ringed player having the ball. He has just beaten the yellow defender and is beyond the two yellow defenders in the x axis.

The Voronoi diagram identifies that there is one player between the attacker and the goal and hence it is a one on one chance meaning that there is a great chance for that player to score.

Other chances include when it is a good time to cross or whether a counter attack is possible. The Voronoi classes run every time a player touches the ball and uses the Game Analyser as well as the Ball class to identify the situation on the pitch and produce comments accordingly.

The Voronoi Class is covered in detail in the Analysis Section.

4.1.4 Memory:

To further increase the complexity of the commentary we had to introduce methods where our system recorded what occurred in the match so far and derive intelligent comments on the game state:

Saved Matches:

This keeps a log of all matches played on our system such as team names and scores. Hence we would comment on past form of the teams and past results if they had met before.

Goals:

The Goals classes would keep a record of who scored and when. This could later be used to determine if a player was on a hat-trick. Lastly, this information would help in deciding the man of the match.

Statistics:

The statistics of the match would be kept in the memory hence during dead moments during the game, the commentator could mention relevant statistical comments about the game. For instance if after a certain period of time in the first half there had not been a shot on goal it could mention that both defences were being solid. Furthermore if one team has a lead of three or more goals the system would mention that it would be a miracle if they lost.

4.1.5 Problems / Outcomes:

There were a few issues with the analysing however we were able to work around the problems and find solutions. One of the main issues occurred in detecting which player had the ball at a current time. The server sends us information when a kick is detected however not all kicks actually touch and move the ball. This meant that in each rule we had to do some error detection to work out the actual last player on the ball. For example, a goal might have been scored by player 11 however the last touch or attempt to kick the ball might have been from the keeper however in our commentary we do not want the keeper mentioned as scoring the goal.

Other issues related to handling the large amount of data being received and enabling efficient analysis of the data in time to be passed to the speech system. Reacting in time to the last data packet received and the use of threads meant the system was able to handle the data without too much delay, therefore complying with the needs of a real-time system.

Last but not least, testing was a problem, as although we introduced new rules for the system to analyse, sometimes the game action was too quick to follow to confirm whether our software had recognised it correctly. To combat this we managed to introduce a Logger Class which would log the events we were looking for into a file. Hence we could replay the matches and check the log to identify whether the analysis was correct.

4.1.6 Further Enhancements:

There are many enhancements that can be added to our system to make it more realistic and efficient. Firstly to ensure maintainability we could use XML rules to conduct the commentary as you have fifteen or so variables which Game Analyser picks up therefore the rules can be written into XML and loaded into the software. While football rules do not change in the real world the simulation has a few changes to rules, adds a few extra and removes some every version.

Secondly keeping past game data on a SQL server would give a more realistic pre-game analysis on all machines. Therefore a central server which could contain more than just the last match results but include the player of the match as well as goals scored by individual players. This would be helpful enhancement for the RoboCup tournaments.

Furthermore our system was based on the commentary system running on a live game however this could be expanded to run on log files of games which have already occurred. This could be helpful for RoboCup tournaments and also for testing of our system.

Large amount of the analysis commentary was on events which had already taken place hence commentary was slow and was produced after the event unlike real commentary which is usually before the moment. Therefore expanding the Voronoi to handle passing would help improve the realism of the commentary. Additionally the Voronoi could be used to determine areas where teams could exploit and hence improve half time and end of match commentary.

Analysing each touch was not perfect as mentioned earlier therefore the system could work out itself whether a player touched or kicked the ball and hence not have to rely on the server protocol therefore giving a less error prone commentary.

Lastly as RoboCup teams use AI techniques to produce players we felt that Neural Networks can also be used to identify which comments are being said analysed regularly and hence ignored but also further to handle new plays and make the commentator more AI and introduce how the commentator can learn through the game and make comments based on AI.

The above enhancements are covered in detail in the Evaluation Section.

4.2 Speech and Atmosphere

4.2.1 Speech Engine:

Selecting and developing an appropriate speech engine for this project was essential to complement the effort that had been dedicated to the game analysis and commentary producing functionality. Keeping in line with other projects in this area, we used an existing text-to-speech engine to enable realistic speech utterance to the audience.

The open-source speech engine selected, FreeTTS^[1], which is purely implemented in Java and can be run directly using Java enabling thorough consistency and efficiency of the calls between the speech producers and the output classes. This engine is flexible enabling us to control the physical parameters of the voice as well as introducing a number of other voices.

The Mbrola synthesizer was introduced to allow two high quality voices to be used as part of our speech interface. Mbrola^[2] plugs directly into FreeTTS without the need for refactoring allowing a simple and concise extension to the code. The voices are used in conjunction with the commentary and stats commentary output, male and female voices respectively.

The engine is controlled using a relatively straightforward interface between the commentary producers and the engine. The commentary (male) voice is implemented in such a way that certain levels of excitement can be parameterised allowing for affective speech. This, which is determined by the game analysers, can be parsed, along with the comment string, to determine and accentuate the tone, tempo, volume and pitch of the voice. A similar idea is implemented in other RoboCup developments such as Byrne, Rocco and Mike^[3] which implement a personalised mark-up language to formalise this parameterisation. We have decided to use the notion of emotions similar to what was implemented in the Byrne system.

The emotion or as we call it, excitation level, is determined by an integer, giving a pointer into the set of parameter constants that respectively set the tone,

^[1] <http://freetts.sourceforge.net/docs/index.php>

^[2] <http://tcts.fpms.ac.be/synthesis/mbrola.html>

^[3] <http://mm-werkstatt.informatik.uni-augsburg.de/files/publications/44/commentator.pdf>

tempo, volume and pitch of the voice that will be used to utter the next sentence. Table 2 below displays the built-in excitation levels available within the release implementation. We have limited the amount of excitation levels to 4 because we did not feel that small increments in parameters were heard on output, therefore we felt that it was sufficient to implement the main levels.

Excite Level	Name	Description
-1	Standard Voice	<i>Defaulted parameters</i>
0	Before Match	<i>Lowered parameters</i>
1	Excited	<i>Increased volume, pitch and tempo</i>
2	Slightly Excited	<i>Small increases in pitch and tempo</i>
3	Goal Voice	<i>Large increases in volume, pitch and tempo</i>

Table 2 – Commentary excitation levels

To enhance the quality and diversity of the commentating, a female voice is introduced. This allows commentating on some of the statistical data collected by the stats functionality. The stats voice is implemented in the same way as the standard commentating voice but without parameterisation as any fractional changes of the female voice parameters causes significant changes in its physical property. To ensure that the male and female voices do not blend, a locking mechanism has been used enabling only one utterance at a time. However, once the block occurs the comments do not queue up, therefore only the most recently parsed comment is spoken. The prioritisation ensures that messages are parsed in time with the events on screen and that the most important comments are spoken.

One of our prime challenges for the speech was to provide ‘affective speech’ as we were limited to using a computer generated voice. We realised that making small modifications led to an important alteration of the physical aspect of the voice, giving the impression that a different commentator was been used. To overcome this problem, we tried to make the voice sound natural by keeping changes to a minimum. Each sentence was spoken with very small, random changes in pitch and tempo. As will be discussed in more detail in 4.2.2 the crowd atmosphere enhanced certain aspects of the voice by reacting to certain events of the match.

As with many integrated automated commentary systems available on the market these days such as EA’s FIFA^[4] series and Konami’s Pro Evolution Soccer^[5], commentary is produced from a pool of recorded sentences from known soccer commentators. This is a more effective way of providing realistic commentary whilst allowing for exceptionally affective speech through such sentences as “Heeennnnrrrryyy” or being able to express anger which was not possible with a text-to-speech engine.

4.2.2 Match Atmosphere

To further enhance the entertainment aspect of this development we decided to introduce samples from crowds around a number of stadiums^[6]. These represent recorded chants, resting crowd noises, goal cheers, general crowd cheers, booing and samples from the referee’s whistle.

These audio samples, used to complement the commentating, are synchronised in time with the match events that develop. The Game Analyser invokes a number of calls within the CrowdManager which acts as a scheduler between all samples. Every call to the CrowdManager dispatches a new sound file to be opened, buffered and played by the CrowdAudioPlayer minimizing lag and keeping synchronized with the match.

All samples are grouped in an array of filenames, according to what they represent. To prevent repetition, we introduce a certain degree of randomness by arbitrarily selecting a sample from each grouping. Each sample is then played out to the audience. To ensure we have continuous broadcasting of samples, we introduced samples that acted as a background loop representing a ‘resting’ crowd. Resting samples are despatched and played continuously by looping from the grouping of filenames, with each iteration randomly selecting the next resting sample.

By allowing the GameAnalyser to call the relevant samples, we were able to synchronise the blow of the whistle, offsidess, goals and near misses with sound. By threading each despatch, we had the ability to enhance performance and synchronisation as well as enabling each sample to effectively blend resulting in realistic crowd atmosphere.

One of our main tests in this area was to provide the continuous flow of crowd samples without there been obvious silences between each. With the steps described above we were able to overcome this problem and provide smooth blending by also introducing fading-in and fading-out of each sample.

Further enhancements could be made through the introduction of further sound samples and effects such as the football hitting the post, or the sounds of stadium announcements after each goal. To enhance the atmosphere further, the selection of chants could be aimed towards a certain team such as, for example, “come on you reds”, although enhancements within this area are limited.

4.2.3 Statistical Analysis and Display

As part of one of our extensions to this project, we decided to provide detailed statistical analysis of match data and display this information through a graphical interface. The statistical data is also utilised by the speech system to generate comments during quieter periods of the match.

Through the entirety of a match, data is collected about goals, shots on and off target, offsidess, number of kicks, average ball position and the number of corners.

^[4]<http://www.fifa07.ea.com/vividas/index.asp?lang=en&tn=1>

^[5]<http://uk.gs.konami-europe.com/game.do?idGame=118>

^[6] Acknowledgments to Championship Manager for the samples. <http://www.championshipmanager.co.uk/>

Although some of this data can be interpreted directly by the audience, we decided to provide means to turn this into information and to provide a graphical interface to render this analysis.

The main user of this output is the GUI which displays the majority of the stats as a formatted list of data as shown in Figure 6. This information is passed directly to the GUI object every two seconds, ensuring that all data on the screen is kept synchronised with the events on pitch. In between this time period the Stats object computes the necessary data to produce information such as possession, which it derives from the number of kicks and ball distribution percentages which is computed from the average ball position.

As basic output of statistical information was seen an essential part of this project, we decided to develop a couple of widgets to complement some of the other statistical data collected. The first widget displays as a 2D line showing the direction and distance of each goal as shown in Figure 5 below.



Figure 5 - InstantReplay Widget displays a goal from player 9, scored from a distance of 20yards

This widget is live throughout the game and the user can interact with it by cycling through each goal by clicking on the button, which will force the window to draw the next line. The information to produce this output is obtained from the GoalDetails class, which adds the parameters of the goal to the set of all goals scored in that match. Parameters include starting and ending coordinates, distance and the player number that scored. By clicking the button, the widget essentially picks the next goal in the set and redraws the window and the goal 2D line representation.

The goal distribution widget essentially displays the percentages of where on average the ball has been, cumulated to the current point in time. This information is computed by obtaining a tally of ball positions at regular match intervals. Percentages are computed through trivial maths and updated to the widget with an interval of two seconds. Figure 4 below displays this widget in action.



Figure 4 - Ball distribution widget

This demonstrates that the ball has spent most of its time in the first third of the pitch, allowing the audience to conclude that the team attacking from right to left has been pressurising the other team.

The main problems encountered in this area revolved around the graphics. Because the RoboCup monitor used a coordinate system that was different to Java's system, we encountered challenges in converting between the two coordinates systems to produce the correct instant replay drawing. The coordinate system in the monitor uses an origin at the centre point of the pitch compared to Java's system where the origin is at the top left of the window. As the monitor reverses the coordinates from negative on left to positive on left at half time, conversion cannot be done efficiently and is error prone. The current implementation does not display the first-half goals at the correct end but does show a perfect reflection through the monitor's origin. However second-half goals are drawn correctly.

As an enhancement to the statistic analysis a match report could be produced showing all the detailed stats as a summary, displayed to screen or printed as a PDF. This is common practice in official FIFA matches {example^[7]} and as it is the aim of RoboCup to create a set of robots to take on human players by 2050 thus we believe that the use of official reporting will be essential.

Common statistical reporting used by broadcasters include the use of player based stats where each player is closely followed on every action throughout a match. This along with more detailed stats such as ball speeds could also be utilised for enhanced commentary and the introduction of a 3D replay system as used by my television broadcasters these days such as the BBC's virtual replay^[8].

4.3 Prioritisation

Text based comments are marked up with variables by the Commentary Producer class, and passed to the Comment Scheduler class where they are added to the proposition pool – a hashtable of all possible comments to be spoken based on game events that have occurred. Every three seconds, the separately executing thread of

^[7]http://us.i1.yimg.com/us.yimg.com/i/fifa/06/w/pdf/64_0709_brl_ita-fra_fulltime__1152478088.pdf

^[8]http://news.bbc.co.uk/sport1/hi/football/fa_cup/virtual_replay/default.stm

the Comment Scheduler selects the comment from the pool with the highest importance rating and passes it to the voice classes for audio output; it also provides a parameter that defines the voice excitement level for that comment. The pool is then emptied, and in three seconds time, the process repeats. The CS can also include a parameter to indicate which of two possible voices will output the speech – there is one for statistical comments and one for game event comments.

Two threads need to access the proposition pool – the thread running the Commentary Producer class to add text comments to the pool, and the separate Comment Scheduler thread to select comments for audio output and delete the pool contents. Due to this fact, access to the hashtable that implements the proposition pool is synchronised to eliminate race hazards associated with shared resources in a multi-threaded environment.

The thread that runs the Comment Scheduler polls continuously to determine when the next comment should be output. It also tests when lower priority comments (the most frequently occurring) were added to the proposition pool, and only outputs them to audio if they were added within the last 100ms – this ensures lower priority audio comments are synchronized with the real time game events taking place on the soccer monitor. Higher priority comments are output regardless of their age in the pool, as a slight lag is preferable to their not being mentioned at all. This lag has been minimised through a number of different iterations of the Comment Scheduler class, along with a number of further optimisations for the most effective, real time audio commentary. For more details, see document: CommentScheduler0.1.

4.3.1 Problems / Outcomes

The final implementation offers a detailed and informative audio commentary that is synchronised with the events taking place on the soccer monitor. All comments relating to high importance events i.e. goals and shots, are output as audio. Less important, high frequency events i.e. passes, loss of ball possession or dribbling with the ball, are not always output as audio. This is due to the fact that the time taken for an audio comment to be spoken exceeds the time duration of the actual event on the soccer monitor. If every comment is output as audio, the commentary will fall behind the game events taking place on the soccer monitor.

The design of each comment and the prioritisation is optimised to give a balance between a richer, more detailed commentary and still providing commentary on the majority of game events. Two voices are used by the system to give the impression of two commentators.

Earlier implementations revealed additional requirements for the scheduling of comments. Without a test to determine when a comment was added to the proposition pool, the highest priority comment was always output. This is desirable in the case of a goal, or high importance event, even if the comment represents an event that occurred in the past. It is not desirable in the case of lower important events, to output an audio comment representing the event if it is not the current game event taking place on the field, as this compromises the real time

delivery of the commentary. By assigning the majority of lower importance game event comments an importance rating of twelve (see document: Comment Priority Levels 0.2), older comments in the proposition pool with the same importance rating are overwritten and thus lower importance comments in the pool are kept as fresh as possible in relation to events taking place on the soccer monitor. Although this defines less of an importance hierarchy associated with comments (see document: Comment Priority Levels 0.1), it significantly improves the real time delivery of the audio commentary and this is the prime concern of the system.

Efficiency is also improved by not adding comments to the pool if they have a lower importance than the comment in the proposition pool with the highest importance rating. As access to the pool is synchronised between threads, the thread that does not have access to the pool is blocked i.e. must wait. By minimising the time a thread may have control of the pool, the waiting time for other threads is minimized, and the speed of access, and thus the system, is increased. This principle also improved when the Commentary Producer class blocked repeated method calls to the Comment Scheduler, relating to the same instance of a game event (see document: CommentaryProducer0.1) as it reduced the number of calls (and requests for a synchronized lock on the proposition pool) made by the Commentary Producer to the Comment Scheduler.

The final version increases CPU usage to around 70%, although it offers a noticeable improvement over the previous versions, in terms of real time synchronization with the soccer monitor.

These are the main outcomes and issues, although the document CommentScheduler0.1 gives further details.

4.3.2 Future developments

Comments have varying lengths of the time taken for the whole comment to be output as audio. The system waits for three seconds (the length of the longest comment) from the start of an audio comment, before outputting the next comment.

This could be improved in future versions by providing the length in milliseconds, of any comment output as audio. The Comment Scheduler could then output the next comment as soon as the current one had finished. More comments could be spoken, but the commentary would be almost continuous. The current implementation produces varying periods of silence between each audio comment, which increases the realism of the commentary. The commentary system is different from most scheduling optimisation problems as having the system process as many tasks (comments) as possible is not necessarily desirable as it would produce a continuous audio output and comments would blur together.

Processing as many tasks as possible is usually the goal of scheduling, such as making optimal use of CPU usage for processing the task queue. Here any idle time

is undesirable and wastes clock cycles.

Also, as a further development, the Comment Scheduler could have an awareness of the ball position on the field, and proximity to players. This could be used to determine if a comment in the proposition pool represents the current game state.

Such functionality would also be very useful in deciding when to output statistical comments. If the ball was being passed around the central area of the football pitch, statistical comments could be output that offer information on player performance, attack patterns, previous game trends, and more.

4.4 Visualisation

The visual function of our project was based around the GUI. The GUI was used as a visual aid for the end-user to view information easily from the match. The GUI contained the text version of the commentary spoken and statistical information regarding the state of the game. The GUI was further extended to display action-replay of the goals from either team and ball position information.

The GUI was developed using two Java libraries AWT (Abstract Window Toolkit) and Swing. Almost everything in our GUI is contained in a top-level window. A top-level window is called a frame and is under the control of the operating system's window management, and which typically can be moved, resized, minimised and maximised independently.

The GUI interacts with three main classes within our project – Game Analyser, Soccer connect and Stats. These enable the GUI to obtain the information to be displayed for the commentary, statistics and play modes. The GUI is updated every second with current information from the match enabling the end-user to keep well informed of various events during the game. Figure 6 below shows the GUI for our RoboCup commentary system.

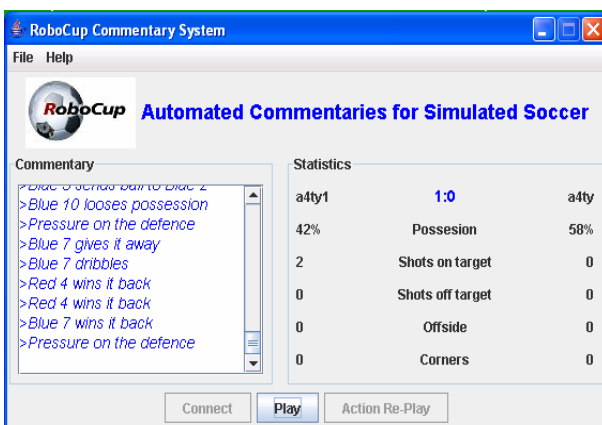


Figure 6 - GUI for RoboCup Commentary System

The GUI is composed of many event and action listeners that respond to end user interactivity. These events of interaction with the GUI involve closing the application, connecting to the server, playing the match, action-reply of the goals and help on using the system.

The objectives that were required for the visual function were met and there were not any major problems with the GUI. However there are some future

developments that could be made to our existing GUI these include the ability to display the commentary in different languages such as in, French, German, Spanish and so on, to enable the user to have more interactivity with the GUI such as having the ability to change the volume, speech in terms of changing the pitch, voice and speed of the commentary. Other developments could involve enhancing the look and feel of the GUI by adding special effects such as a splash screen and a background image. A graphical enhancement to our system can be based around the research we undertook from reading about the Byrne commentating system, is to implement a similar feature in our system that would allow the user to view an image of someone speaking to them, rather than listening to background commentary playing while the game is in action.

Overall the requirements for the visual function were met successfully.

5. Conclusion

In conclusion, we created an automatic commentary system with real-time speech synthesis for the RoboCup's Soccer Server. We described the Soccer Server that follows the play of a game, and explained how their output is combined with our system to produce a commentary using the four main functions described earlier in the report. Our system follows the ball-by-ball action with commentary scheduling allowing time for giving information on the global play of the teams. In making these global analyses, it brings to bear information from databases of previous matches and also its knowledge of teamwork and soccer.

As discussed above in the report our project has the capability to be further extended in more ambitious directions, however with a lack of time these have not been implemented. However we have reached our goal of delivering a system that automatically generates commentaries for RoboCup games, and plays these alongside the game in real-time using speech synthesis. The success of our system shows that RoboCup is not just a robot competition – it is a challenging domain for a wide range of research areas, including those related to real-time natural language commentary generation.

6. Acknowledgements

We would like to thank Colin Johnson for the help and guidance provided throughout the development of this project.

We would like to thank a4ty [F] for the use of the players that has been used to test our system.

7. References

- [A] RoboCup Official Site <http://www.robocup.org/>
- [B] Research paper – Rocco, Byrne and Mike <http://www.cs.gmu.edu/~sean/papers/commentator.pdf>

- [C] Rocco research paper
<http://www.dfki.de/imedia/robocup/>
- [D] Byrne research paper
<http://www.cs.gmu.edu/~sean/papers/byrne.pdf>
- [E] Voronoi diagrams of semi-algebraic sets
<http://tel.archives-ouvertes.fr/docs/00/04/78/27/PDF/tel-00005932.pdf>
- [F] Reference to a4ty
<http://www.cs.rtu.lv/dssg/en/research/robocup/Default.htm>
- [G] Barnes, D. Object Orientated Programming with Java,
Prentice Hall,