A Team of Humanoid Game Commentators

Manuela Veloso, Nicholas Armstrong-Crews, Sonia Chernova, Elisabeth Crawford,
Colin McMillen, Maayan Roth, and Douglas Vail
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA

***Abstract*— We present our work on creating a team of two humanoid robot commentators for soccer games of teams of four AIBO robots. The two humanoids stand on the side lines of the field, autonomously observe the game, wirelessly listen to a "game computer controller," and coordinate their announcements with each other. Given the large degree of uncertainty and dynamics of the robot soccer games, we further introduce a "puppet master" system that allows humans to intervene in a sliding autonomy manner, prompting the robots to commentate on an event if undetected. The robots process then input from these three sources, namely own and shared vision, game controller, and occasional puppet master, to recognize events which they translate into a varied set of predefined announcements. We present the behavioral architecture, the vision-based event recognition, and the game-based adaptive criteria for the selection of comments. We exemplify our development with multiple illustrative cases corresponding to different game situations. In summary, our work contributes a team of two humanoids fully executing a challenging observation, modeling, coordination, and reporting task.**

***Index Terms*— Entertainment; Gesture and communication; Cooperating humanoids**

I. Introduction

Research in robot soccer has rapidly progressed in the last ten years [1], [2], [3]. The robots successfully compete as teams, perceiving a challenging dynamic environment, making decisions, cooperating as a team, and acting to achieve concrete objectives. Although the robots play the game, humans perform all the other functions associated with the game, including being commentators and referees. So one question arose on whether we could develop robot commentators and referees. In this paper, we present how we address the commentator task with a team of two humanoid robots, namely two Sony QRIO robots [4].[1] We aim at a future extension to robot referees and even coaches.

We develop the commentators for the specific game of the AIBO robot soccer games, where teams of four AIBOs compete. Our two robot humanoid commentators stand on the side line of the field, follow the game, and announce game events. Figure 1 shows the setup.

Each humanoid robot commentator has a stereo vision camera, onboard computing for processing its perception, cognition, and motion, multiple sensors and actuators, and wireless communication. The robots observe the AIBO game. They assess the state of the world from three different sources: (i) their own vision, (ii) a computer game controller that records and transmits the calls of the human referee, namely goals, fouls, and ball out, and (iii) a "puppet master" as a QRIO controller that can be activated by a human to prompt the commentator to announce any event that they may not have detected through their vision or may not have been called by the referee.

[1]QRIO were developed by Sony. Sony finished any new developments of QRIO as of now.

Fig. 1. Two QRIO Robot Commentators for an AIBO Robot Soccer Game.

There are several previous approaches that demonstrated soccer commentators for either real soccer images ([5]), or for simulation soccer([6]), or for the small-size RoboCup soccer game([7], [8]). Except for one of these efforts ([7]) with a humanoid head, the commentators were not done with humanoids, showing that the core task of commentating a game does not necessarily "need" to be performed by a humanoid robot. We present two humanoids performing this task for the RoboCup AIBO game, which builds upon previous research but departs from it for the first time for this RoboCup league. We choose this task for our humanoids for two main reasons. Firstly, we view the commentator task with an additional goal of interaction with the audience, which fits well the use of humanoids. The interaction, as we have developed so far, can be viewed as a one-way interaction with the audience, as the robots behave to announce and entertain, but they do not process visual or sound information from the audience. Creating the full two-way interactive commentators is one next step for future work. Secondly, as it is hard to find tasks for full humanoids to autonomously perform, we find that this commentator domain provides a concrete challenge for the robot humanoids, requiring them to completely and robustly integrate their perception, cognition, and body motion. An additional interesting aspect of our work, which further

distinguishes it from previous robot commentator efforts, is our use of two humanoid robot commentators. We pursue research on multi-robot systems, and the fact that the playing field is larger than the range of the vision of a single QRIO, offered a great opportunity for us to investigate a team of two humanoid commentator robots. Finally, although our work is developed within the specific commentator task, we aimed at contributing a general architecture and algorithms potentially capable of being used in other similar "observation-reporting-motion" multi-robot tasks. In the paper, we present the observation, control, motion, and interaction components within the commentator domain while pointing also the underlying general contributions.

We organize the paper as follows. Section II addresses the robot behaviors. It first presents the overall behavior architecture introducing the connections between the reasoning and the multiple sources of input and output. It then introduces the complete behavior algorithm that allows the robots to robustly identify and call events in the game. Section III and IV respectively present the recognition of events through vision processing and from wireless communications, namely from the game computer controller and the puppet master systems. Section V presents the library of announcements with multiple versions of comments for the same event. It further explains the algorithm for selecting announcements as a function of the run of the game. Finally Section VI draws conclusions on the work presented.

II. Reasoning

At a high level of abstraction, the commentator task is clear: the commentators observe the game, recognize events, and convert their recognition to audible announcements. Our behavior architecture captures the interactions between these three underlying main concepts: observations, events, speech.

A. Overall Behavior Architecture

Figure 2 shows a high level overview of the overall architecture of our system, CMCast, consisting of the two robots, a centralized control module, the Director, and two external input sources, i.e., the Game Controller and the Puppet Master.

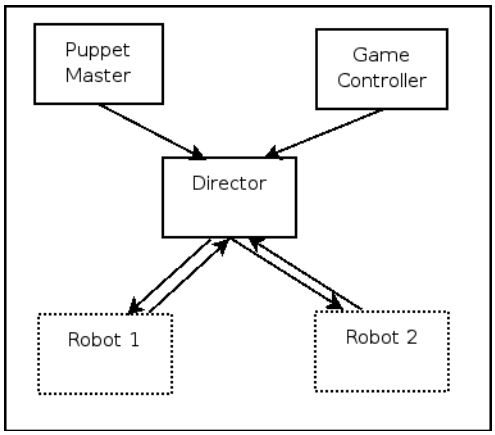

Fig. 2. The CMCast Overall Architecture.

The commentator task sets a clear requirement for quite immediate response to events. Although the robots have onboard computation and behaviors, as we present below, the potentially delayed wireless communication among them, prohibits from having completely distributed control. We therefore chose to include the centralized Director module, responsible for processing external input and guiding the behaviors and coordination of the two robots.

Input to the Director comes from three classes of sources: the Game Controller, the Puppet Master, and the robots, in our case two. The Game Controller is the computer referee of the robot soccer four-legged AIBO league. The games have a human referee who verbally makes the calls. As the AIBOs do not process the speech of the referee, the calls are entered by another human game official into the Game Controller, which in turn wirelessly sends the calls to the AIBO robots. The robot players can then act autonomously in response to the called game situations, such as kickoff, balls out, and fouled or penalized robots. The Game Controller became an obvious source of input for the robot commentator task, as it captures the set of called game events, which can then be passed wirelessly to our CMCast Director.

We envisioned CMCast commentating on a larger set of events than the ones directly available from the Game Controller. As we present below, by following the position of the ball on the field, the robots are able to recognize a set of events related to the flow of the game, such as interesting kicks. However, even with the combination of the Game Controller and the events detected by their own vision, we knew that there could be interesting events that would not be detected.

We created an additional interface to input to the Director, the Puppet Master, in which we can manually provide additional game state information that is relevant to the commentator task but not easily sensed by the robots. Section IV discusses in detail the Game Controller and Puppet Master.

Finally the Director interacts with the robots. It receives input from the robots consisting of data resulting from the robot vision processing, as presented in Section III. Additionally, each robot reports the termination of execution of each motion command.

The Director collects and maintains its input information in a game or "Event History," and uses it centrally decide on the robot behaviors. The Director sends to the robots coupled speech and primitive behavior commands. The primitive behavior commands are sent to each of the robots, where they are executed by the onboard Behavior algorithm. Through the input of the robots' report on the completion of each command, the Director synchronizes the actions of the robots both with respect to each other and the speech generated by the offboard Text-To-Speech system. In the current implementation of CMCast, speech is generated offboard in order to allow the use of loudspeakers in the noisy stadium environment typical of this domain. The system can be easily modified to use onboard speech synthesis instead.

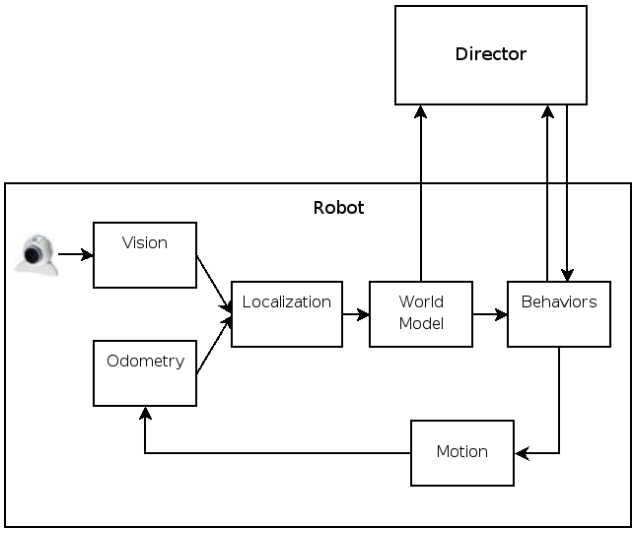

Fig. 3. The CMCast Robot OnBoard Behavior Architecture.

B. Onboard Single Robot Behavior Architecture

Figure 3 shows the behavior architecture onboard the robots. A series of modules form the main pipeline, in which data is processed in each step of the sequence.

The reasoning begins with the input from the robot's onboard camera images, from which object data is processed by the "Vision" processing. "Localization" determines the location and orientation of the robot using the distances to field landmarks and simple humanoid robot "Odometry" models. The "World Model" uses the robot and ball position relative to the robot to represent the location of the ball in terms of its global position, which is passed by each robot to the Director. The onboard Behavior module executes primitive behaviors, as instructed by the Director, using world state information available from the World Model. Primitive behaviors include the execution of prescripted gestures, ball tracking and ball search, all of which are well suited for onboard behaviors, as they require low latency for smooth execution.

C. Director Behavior Algorithm

During execution, the Director records all of its inputs in the Event History, which maintains all information about the game state, such as the score or which robots are penalized. Additionally, we include in the Event History processed game statistics of relevant historical information, such as counts of penalties for each team and robot. CMCast uses this historical information to make interesting comments on the development of the game over time; formally, it transforms the non-Markovian process over game state information into a Markovian process over the Event History variables.

For each iteration of the algorithm, the Director module evaluates its input, detects significant events that may have occurred, and triggers the appropriate responses for each of the robots. The resulting output can vary between a single utterance or motion, to a full dialog between the two robots. Section V describes the algorithm for selecting the appropriate response to a detected event.

III. Event Recognition Through Vision Processing

Each QRIO robot is equipped with the same color camera which is used on the AIBO robots. We perform color segmentation on these images using the CMVision [9] image library and then run a series of object detectors on the segmented images to determine which objects are present in the images. Figure 4 shows an example of a raw and the corresponding color segmented image.

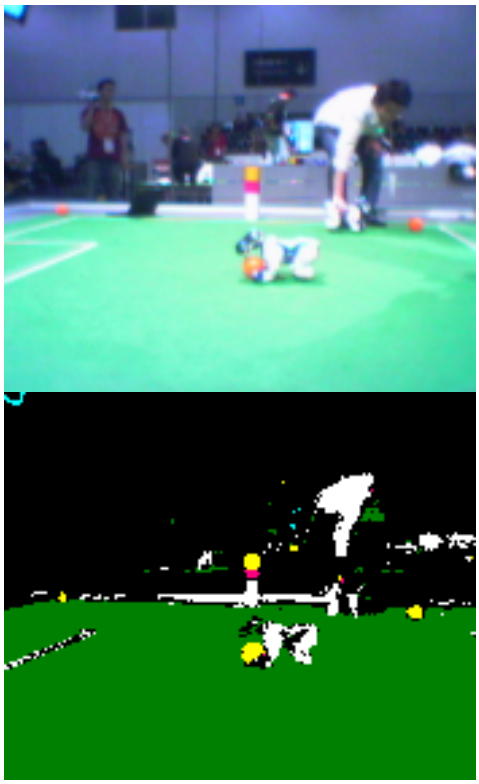

Fig. 4. An example of a raw image from the robot's camera (top) along with the corresponding color segmented image (bottom), used for object detection. In the images, an AIBO robot in a blue uniform holds the ball under its chin in preparation for a kick. A localization marker, consisting of two colored bands on top of a white column, is visible at the edge of the field.

In addition to detecting the presence of objects of interest, these object detectors also return an estimate of each object's position as well as a heuristic confidence indicating how likely it is that the object is truly present. Object detectors look for: the ball, goals, robots, and localization markers.

Vision on the robot serves two purposes. The first of these is to provide low latency information to onboard behaviors so that the robot can quickly and realistically respond to events as they unfold. In the commentator domain, this mainly takes the form of ball tracking; the ball position estimates returned from the vision module are used to servo the robot's head and body to make it clear to observers that the robot is paying attention to the current state of the soccer game. In addition to ball tracking, information from vision of localization beacons is used by the robot to compute its current position so that object

positions can be translated into a global coordinate frame and shared between robots. Additionally, vision of the goals is used by the robots so they can focus on the appropriate goal in response to events such as points being scored.

The second purpose of vision is to detect events for the robots to incorporate into their commentary. Currently, two types of events are detected: the presence of the ball in a particular region of the field and times when the ball has been kicked by a robot. To detect the first type of event, the robot uses its ball position and localization estimates to determine when the ball lies in an interesting region of the field, such as the areas near either of the goals. To determine whether or not the ball has been kicked, the robot keeps a running history of ball positions. The movement vectors between the positions in this history are compared and a heuristic confidence is computed based on the length of the vectors as well as their co-linearity. In practice, a history length of six ball sightings, or one half second's worth of data, was sufficient to detect when the ball was kicked.

IV. Event Recognition from Wireless Sources

The QRIO robots are equipped with 802.11b wireless LAN cards. Wireless networking allows the robots to communicate with each other; the wireless capabilities also enable the robots to recognize events from two wireless sources of input: the Game Controller and the Puppet Master.

A. Game Controller

The referees of the four-legged league use a software program called the "Game Controller," which wirelessly sends referee calls and similar information to the players. Figure 5 shows the Game Controller interface. Our commentators also listen to this data source, which allows them to detect a rich class of events that would be difficult or impossible to detect through vision. Specifically, the Game Controller provides the following information to the robots:

- Game state: the current score, whether the game is in the first or second half, the time remaining in the half of the game, and whether a team is about to kick off.
- All the penalties that can be called against the robots: ball holding, illegal defender, goalie pushing, player pushing, leaving field, pick-up request, illegal defense, obstruction, and damage. The penalty data tells us which robot(s) were penalized for each foul.
- "Ball out" calls made by the referees.
- Time-outs called by either team.

From the Game Controller data, our robots detect several different types of events. They comment on each type of penalty by calling the foul, such as "*That was a player pushing penalty on Red.*" The first time each penalty is called in each half, one of the two commentator robots explain the rule in more detail. This rule explanation serves the important role of introducing the audience to the probably unknown robot soccer rules. For instance, the first player pushing penalty would cause our commentator robot to explain: "*Players cannot push each other for more than three seconds.*" Players that are penalized are removed from the game for 30 seconds. We can use this information from the Game Controller combined with the Event History to account for situations when more than one that robot are penalized. The robots then comment when a team has many penalized players; e.g., "*Many Red players are now out of the game. This is a big opportunity for Blue!*"

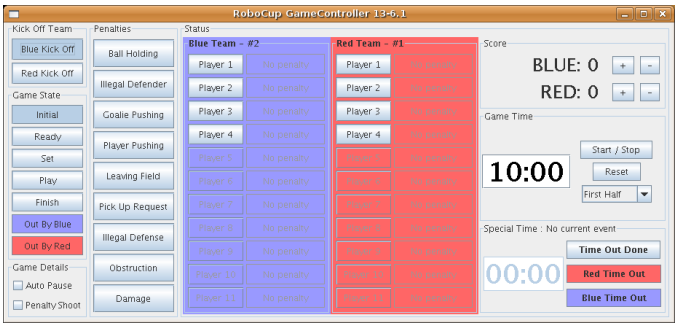

Fig. 5. Screenshot of the Game Controller interface.

Goals are also detected from the Game Controller data. The specific commentary used for a goal depends strongly on the past history of goals scored. Goals are announced differently sensitive to the specific sequence in which the current score was reached, including whether it's the first goal of the game, when one team is far ahead of the other, when one team seems to be making a comeback, or when the score is equalized.

Other events detected from the Game Controller data include ball-out events and the time remaining in the half of the game being played. Reporting on the time remaining is particularly important, since the official game clock is not typically visible to the audience except during the final games.

B. Puppet Master

The Game Controller provides notification of many events that the robots may otherwise have been unable to recognize autonomously. However, there are still a significant number of interesting events that are not detectable either with the onboard vision or the Game Controller input. For instance, the AIBOs sometimes crash due to empty batteries or software errors. It would be difficult for the commentators to visually recognize a crashed robot (as opposed to a stopped robot). For situations like these, we have developed the offboard Puppet Master controller to allow a human operator to artificially insert specific types of events, such as robot crashes, into the commentators' event history. Figure 6 shows a screenshot of the Puppet Master interface.

The Puppet Master represents the interesting feature to complement the autonomous perceptual, reasoning, and motion behavior of the robots if needed. The Puppet Master consists of three main functional parts, namely:

- **Call for known undetectable, interesting events** - As shown in the top part of the interface, the Puppet Master includes choices to manually select several events that we know may be part of the game, but we also know that the robots cannot detect them, as of now. There are three types of such events:

– *Predefined events*, such as a Pass or a Nice Save, which we have associated with predefined announcing speech and motion. Our robots also do a special sequence of actions for a half-time show, and at the beginning and end of the game.
– *"Filler" events*, that we use to fill in if nothing really interesting is happening. CMCast includes a set of predefined "filler" comments that can then be used. When the filler event is invoked in the Puppet Master, the robots comment on something not directly related to the current game state, such as further explanation of the game rules, the abilities and hardware of the AIBO robots, and team-specific details such as their team captain, areas of research interest, and results from past RoboCup competitions.
– *Concrete speech and motion invocations*, which can be entered manually if they are not part of our set of predefined events (as shown under the "Say", "Motion" empty fields at the top of the interface). The more the robots do autonomously, the less we use this feature. (In fact, in the commentating of the multiple games at the recent RoboCup'2006 event, we never used this feature during the games.)

- **Perceptual guidance** - Our CMCast robot commentators are positioned in rather fixed positions on the side line of the field (see Figure 1). They can slightly move their bodies but not in a way that allows them to position themselves freely to track the ball. The ball is therefore not always visible by either of the two robots. Furthermore, the ball's position "jumps" as it is manually positioned by the human referee in different locations on the field after a ball-out event. Through this feature of the Puppet Master, we can give guidance on the ball position so that the robots actively direct their heads to the entered ball location. The input is given at a high-level of granularity, as we discretize the field into 25 cells, as shown in Figure 6, with the two blue (B) and yellow (Y) goals.
- **Motion guidance** - Finally, the robot's announcing motion can make the robot slide away from its desired position, namely outside of the field and close to the line. As the robots' localization or odometry may have modeling errors, we need to ensure that the robots stay outside the field boundaries, such that they do not interfere with the normal game play. This feature of the Puppet Master, as represented by the multiple arrows, acts as a remote motion controller to the robot, when needed.

In general, the Puppet Master represents a sliding autonomy approach. In theory, and if humanoids will one day have a complete perceptual understanding, including speech and vision, this feature would disappear. However this is not the case yet. Therefore the interest in the Puppet Master.

V. Commentating - Speech and Gesture

The commentary given by the QRIOs is event driven. Interesting events are detected from the current game play

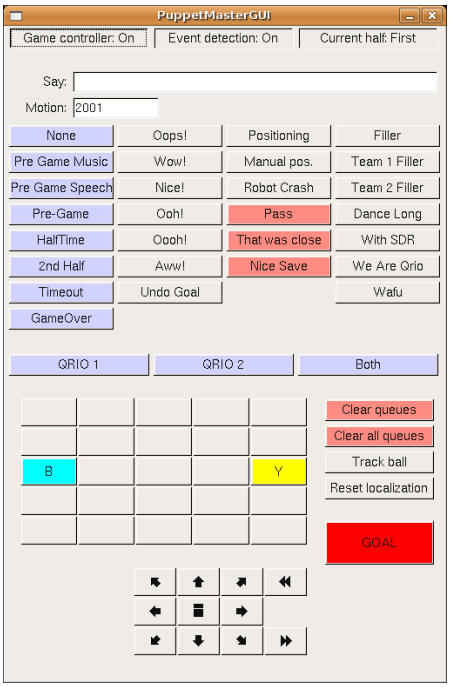

Fig. 6. Screenshot of the Puppet Master interface.

history, and are used to generate the actions of the QRIOs. The commentator algorithm processes the events and decides what the QRIOs should say and how they should move. The aim of the commentator algorithm is to ensure that the QRIO commentators act in a timely, informative and entertaining way.

The commentator algorithm listens to the vision, the game controller and the puppet master in order to receive information about what we call *base-events*. Base-events represent immediate occurrences in the game like fouls, goals and exciting kicks. Whenever a base-event is received by the commentator algorithm, a set of predicates are executed to determine whether or not a *special-event* should be generated. Special-events are generated based on the history of the game. For instance, a special- event is generated when a team scores and becomes 3 goals ahead in order indicate that one of the teams is dominating the play. In some cases when a special-event is generated the base-event is not acted on and in other cases neither the base-event or a special-event are acted on. The use of special-events allows the robot's actions to be adapted to the history of the game. We call the union of base-events and special-events *Events*.

In order to decide whether to act on a particular Event, the commentator algorithm tracks when each Event was last acted upon. Each type of event has a *cool-down* period associated with it that specifies the time the QRIOs must wait between acting on the event. This helps to keep the commentary varied. If an Event has not been acted upon recently it is added to the queue of events to act on. The commentator algorithm continually removes the oldest Event from the queue and selects an Action for the QRIOs to take based on that Event.

The queue of events is kept very short in order to ensure that the commentary stays up-to-date. If a new Event arrives

and the queue is full, an old item is removed unless it relates to a very important event such as a goal. Certain events, which are not particularly important, and only suitable for acting on right away, are never enqueued when the QRIOs are already acting on another event. An example is the vision generated Event that one of the teams is making good progress on the field. This is not added to the Event queue unless the queue is empty and no action is currently being executed. Goals are dealt with in a special way in order to ensure they are acted upon in a timely manner. When a goal occurs and the current action is interrupted and the Event queue cleared of all non goal related events.

Because the aim of the CMCast QRIO commentators is to be entertaining as well as informative, it is not enough for each Event to generate a single output behavior on the part of the robots. Instead, we introduce a library of *Actions* and allow for a one-to-many mapping of Events to Actions, keeping the robot behavior from becoming repetitive. Each Action is composed of an *Utterance*, which is the verbal statement that the robot says, and a *Gesture*, an accompanying physical motion. Actions may be performed individually by a single robot, jointly by both QRIOs, or may be composed of a sequence of Utterances and Gestures forming a "conversation" between the two commentators. For example, there are many possible Utterances that the QRIOs may say when a goal is scored, from the simple "Goal!" to the more elaborate "A great goal for the Blue team!"

Each processed Event corresponds to a set of several Actions. An action-selection mechanism is needed to ensure that a diverse set of behaviors results from Events that occur multiple times in a single game. We introduce an action-selection mechanism with two components, a cool-down function and a max-usage rule. Each Action is assigned an initial weight, and Actions are selected probabilistically on the basis of these weights. When an Action is selected, its weight is lowered, reducing the likelihood that it will be selected in the near future. The cool-down function raises Action weights over time. The rate of increase and the maximum weight for each Action may be inputted with each Action definition. Each Action may also be assigned a max-usage rule, limiting the number of times that it can be selected to a fixed constant. Once the maximum number of usages has been exceeded for a particular action, it is no longer eligible to be selected.

Table I summarizes the commentator algorithm in pseudo-code. The method getEvents denotes the process previously described whereby the history is used to process the base event and a special-event is possibly generated. The selectAction method selects actions probabilistically based on their weight as we have have already described.

```
procedure eventListener:
1. when a base-event occurs:
2.    current-events = getEvents(history, base-event)
3.    for event in current-events:
4.       if event has not been acted on recently:
5.          processEvent(event)

procedure processEvent(event):
1. if event == Goal:
2.    clear events-queue and act on the event immediately
3. else if robots currently acting:
4.    evaluate if the event should be added to events- queue
5.    remove old events from events-queue if it is maximum size
6. else robots not already acting: (events-queue is empty)
7.    action= selectAction(Event)
8.    perform-action(action)

procedure performAction(action):
1. send speech and gesture commands for action
2. once action has executed, if events-queue not empty:
3.    action = selectAction(events-queue.pop())
4.    perform-action(action)
```

TABLE I

A SUMMARY OF THE ANNOUNCING (SPEECH AND GESTURE) ALGORITHM

VI. CONCLUSION

The commentator task is an interesting domain for humanoid robots. We present our work on first humanoid commentators for the AIBO robot soccer games. The game is played on a large field, is highly dynamic, and is hard to completely apriori model. We have presented two robot QRIO commentators that detect a large set of events recognized from a computer game controller, a puppet master, and the robots' own vision. Our complete system, CMCast, is fully implemented and was demonstrated at recent RoboCup 2006 event in multiple robot soccer AIBO games. The robot commentators successfully and autonomously observed the game and announced the events through varied utterances and motion adapting to the sequence of the game and the different teams. [2]

REFERENCES

[1] M. Veloso, W. Uther, M. Fujita, M. Asada, and H. Kitano, "Playing soccer with legged robots," in *Proceedings of IROS-98, Intelligent Robots and Systems Conference*, Victoria, Canada, October 1998, pp. 437–442.

[2] H. Kitano, M. Fujita, S. Zrehen, and K. Kageyama, "Sony Legged Robot for RoboCup Challenge," in *Proceedings of ICRA-98, the 1998 IEEE International Conference on Robotics and Automation*, Leuven, Belgium, 1998, pp. 2605–2612.

[3] G. Lakemeyer, E. Sklar, D. Sorrenti, and T. Takashi, Eds., *RoboCup-2006: Robot Soccer World Cup XX*. Springer-Verlag Press, 2007, forthcoming.

[4] M. Fujita, K. Sabe, Y. Kuroki, T. Ishida, and T. D. Toshi, "SDR-4XII:A Small Humanoid as an Entertainer in Home Environment," in *Robotics Research: The Tenth International Symposium*. Springer Tracts in Advanced Robotics, 6, 2003.

[5] E. André and G. Herzog and T. Rist, "On the simultaneous interpretation of real world image sequences and their natural language description: The system soccer," in *Proceedings of the Eighth ECAI*, Munich, 1988, pp. 449–454.

[6] E. André and K. Binsted and K. T. Ishii and S. Luke and G. Herzog and T. Rist, "Three RoboCup simulation league commentator systems," *AI Magazine*, vol. 21(1), pp. 57–66, Spring 2000.

[7] I. Frank, K. Ishii, H. Okuno, J. Akita, Y. Nakagawa, K. Maeda, K. Nakadai, and H. Kitano, "And the fans are going wild! SIG plus MIKE," in *RoboCup-2000: Robot Soccer World Cup IV*, P. Stone, T. Balch, and G. Kraetzschmar, Eds. Berlin: Springer Verlag, 2001.

[8] K. T. Ishii, I. Noda, I. Frank, H. Nakashima, K. Hasida, and H. Matsubara, "MIKE: An automatic commentary system for soccer," in *Proceedings of the Third International Conference on Multi-Agent Systems*, Paris, July 1998, pp. 285–292.

[9] J. Bruce, T. Balch, and M. Veloso, "Fast and inexpensive color image segmentation for interactive robots," in *Proceedings of IROS-2000*, Japan, October 2000.

[2]We thank SONY for making the QRIOs available to us for this task. We further thank them for their multiple developed motion and software features that underly our developed CMCast system.