

A TEAM OF HUMANOID GAME COMMENTATORS

MANUELA VELOSO, NICHOLAS ARMSTRONG-CREWS, SONIA CHERNOVA,
ELISABETH CRAWFORD, COLIN McMILLEN, MAAYAN ROTH,
DOUGLAS VAIL, STEFAN ZICKLER

*School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA*

Received Day Month Year
Revised Day Month Year
Accepted Day Month Year

We present a team of two humanoid robot commentators for AIBO robot soccer games. The two humanoids stand on the side lines of the playing field, autonomously observe the game, wirelessly listen to a “game controller” computer, recognize events, and select announcing actions that may include coordination with each other. Given the large degree of uncertainty and dynamics of the robot soccer games, we further introduce a “puppet master” control that allows humans to intervene, prompting the robots to commentate an event if previously undefined or undetected. The robots recognize events based on input from these three sources, namely own and shared vision, game controller, and occasional puppet master. We present the two-humanoid behavioral architecture and the vision-based event recognition, including a SIFT-based vision processing algorithm that allows for the detection of multiple similar objects, such as the identical robot players. We introduce the commentating algorithm that probabilistically selects a commentating action from a set of weighted actions corresponding to a detected event. The probabilistic selection uses the game history and updates the action weights to effectively avoid repetition of comments to enable entertainment. Our work, corresponding to a fully implemented system, CMCast, with two QRIO robots, contributes a team of two humanoids fully executing a challenging observation, modeling, coordination, and reporting task.

Keywords: Cooperating humanoids; Controlled autonomy; Event recognition; Multi-object visual detection; history-based probabilistic action selection.

1. Introduction

Research in robot soccer, in particular within RoboCup, has rapidly progressed since its beginnings in the mid 1990s.^{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. One challenging question is whether we can also develop robot commentators and referees for the game of robot soccer. In this article, we present how we address the commentator task with a team of two humanoid robots,

namely two Sony QRIO robots.^{4,a} Our work enables future extensions to robot referees and even coaches.

We developed the first humanoid robot commentators for the specific game of the RoboCup Four-Legged Robot League, in which two teams of four Sony AIBOs compete. There are several previous efforts 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 which used a humanoid head, the commentators were not done with actual mobile humanoid robots, showing that the core task of commenting a game may not necessarily “need” to be performed by a humanoid robot.

We choose the commentator task for our humanoids, however, for two main reasons. Firstly, we view the commentator task with an additional goal of interaction with the audience, which fits well with the use of humanoids. The interaction that we have developed so far can be viewed as a one-way interaction with the audience: the robots act to announce and entertain, but the robots do not process visual or sound information from the audience. Creating 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 humanoid robots, 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 offers 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 aim at contributing a general architecture and algorithms potentially capable of being used in other similar “observation-reporting-motion” multi-robot tasks.

Our two robot humanoids act as a team of commentators, standing on the side line of the RoboCup AIBO playing field, following the game, and announcing game events. Figure 1 shows the setup.

Each humanoid robot commentator has a stereo vision camera, on-board 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 “game controller” that transmits the calls of the human referee, such as goals, fouls, and balls going out-of-bounds, and (iii) a “puppet master,” a control interface that allows a human to prompt the commentator to announce any event that the robots may not have detected through their vision or that may not have been called by the referee.

^aThe QRIO robots were developed by Sony. Sony has since ceased new developments on the QRIO robot platform.

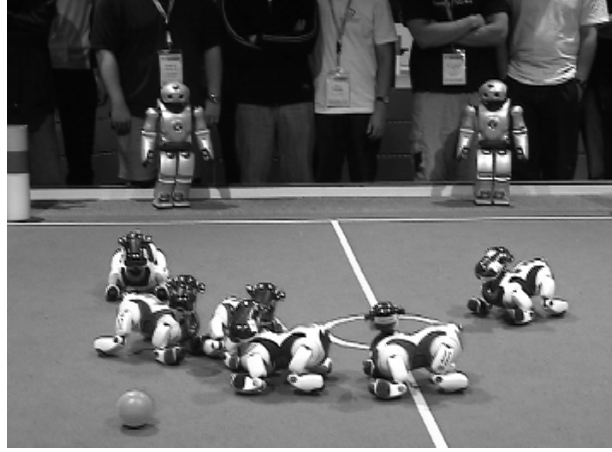


Fig. 1. Two QRIO robot commentators for an AIBO robot soccer game.

We organize the article on our team of humanoid game commentators as follows. Section 2 addresses the robot behaviors. This section 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 3 presents the vision processing to enable event recognition. Game objects are processed based on color. A SIFT-based vision algorithm enables visual recognition of multiple objects, such as the AIBO soccer players.^b Section 4 then presents our event recognition from wireless communications, namely the referees' game controller and the "puppet master" control interface. Section 5 presents the library of announcements, which allows multiple speech and gesture commands to be generated for similar events. It further explains our algorithm for selecting announcements as a function of the game history. Finally Section 6 draws conclusions on the work presented.

2. Architecture

At a high level of abstraction, the commentator task consists of a loop in which the commentators 1) observe the game through various sensory and wireless input sources, 2) filter those inputs to recognize salient events, and 3) provide event announcements and relevant commentary by means of speech and gestures. Our behavior architecture captures the interactions between these three underlying main concepts: observations, events, and actions.

^bWe did not include this autonomous multi-object detection feature in our demonstrated CMCast commentator system at RoboCup 2006, mainly for lack of development time. However, the algorithm has been successfully demonstrated using real video footage of RoboCup games. The robots in the demonstrated CMCast announced the game based on visual tracking of the colored ball.

2.1. 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.

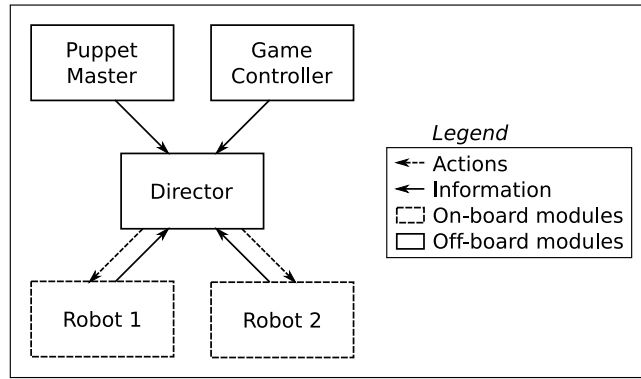


Fig. 2. The CMCast overall architecture.

The commentator task sets a clear requirement for immediate responses to events. To ensure this capability, a centralized decision module the Director, is responsible for processing external input and coordinating the behavior of the two robots. Centralized control enables fast and efficient decisions without the need for decision-making negotiation strategies over the wireless network. The Director chooses the robots' actions, which are executed by their on board processors.

Input to the Director comes from three classes of sources: the Game Controller, the Puppet Master, and the two robots. The Game Controller is the referee interface used within the RoboCup four-legged robot league to communicate referee decisions to the robots. All calls within a game, such as kickoff, balls out, and penalties, are made verbally by a human referee who observes the game. The calls are entered by another human game official into the Game Controller, which in turn wirelessly communicates this information to the AIBO robots. The robot players then act autonomously in response to the called game situations.

Given that this Game Controller exists basically to transfer human referee commands to the robot players, we introduce it also as an input source of information to the Director. The Game Controller provides valuable and precise information about the current state of the game, which can not be always determined through other means (e.g., the robot commentators can not determine which robot was penalized using vision because all AIBO robots on a team have the same visual appearance). The Game Controller enables the commentators to report the state of the game and to explain the various rules of the game, such as penalties, as they occur.

Not all interesting events have human referee calls associated with them. Therefore not all events, on which the robots would need to commentate, are available from the Game Controller. For example, when a particularly good shot is made by an attacker, or a goalie robot makes a spectacular save, the commentators should respond to these events, even if these events do not result in a referee call. We resort to recognized some of these events through the robots' on-board vision, as we describe in Section 3. This vision-based event information is reported by the robots to the Director module.

The combination of the Game Controller and the robots' own vision to detect a series of predefined event classes will inevitably still not be able to detect "all" interesting events. It may be that the robot behavior is difficult to recognize through vision processing, that the event occurs out of view of the robots' cameras, or that an exceptional circumstance occurs that was not included in the predefined possible set of events. We introduce another input to the Director module, the "Puppet Master," which provides a human commentator assistant with a means to manually supplement event data. Section 4 discusses further details of the Game Controller and Puppet Master.

Given the three types of sources of input – Puppet Master, Game Controller, and Robots – (see Figure 2), the Director collects and maintains a *Game History*, as a set of game statistics that are relevant for later announcements. For example, the Director keeps a count of penalties for each robot so that it can announce when a particular robot has suffered an additional penalty. In each iteration of the algorithm, the Director evaluates its input, detects significant events that may have occurred, and triggers the appropriate commentating 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 5 provides further details about the game history and also describes the algorithm for selecting the appropriate response to a detected event. Commands for physical motions, such as gestures, are communicated to each robot from the Director via the wireless network. Each robot executes its specified commands using an on-board behavior module, as we now describe.

2.2. On-board Single Robot Behavior Architecture

Figure 3 shows the QRIO robot on-board behavior architecture. Processing begins with the image data acquired from the robot's camera, from which information about objects in the robot's environment is extracted by the Vision module. The position and orientation of the robot is calculated by the Localization module, based on the observed locations of fixed localization markers (colored beacons placed around the edges of the AIBO playing field that act as landmarks), as well as the robot's Odometry model. The World Model module uses the vision and localization data to track the global positions of other objects detected by the robot in the environment, such as the ball. The global position of the ball is communicated

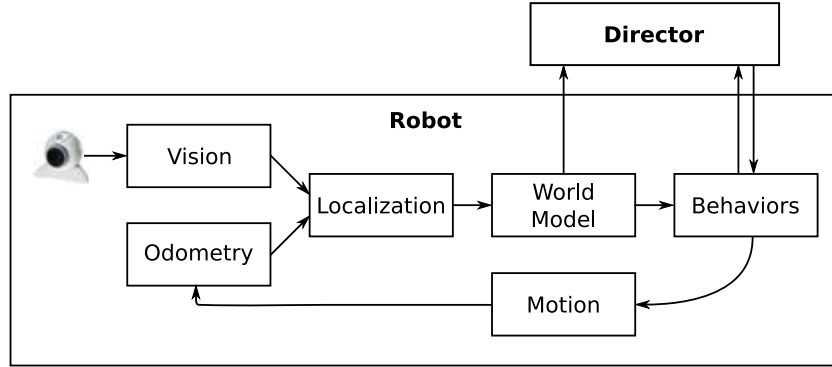


Fig. 3. The CMCast robot on-board behavior architecture.

by the robot to the Director, enabling both commentators to maintain up-to-date information even if one robot may be observing another area of the field. Finally, the Behavior module uses this information to execute primitive behaviors as instructed by the Director.

The set of primitive behaviors includes motion gesture commands, ball tracking behaviors, and ball search. All of these behaviors require low latency for smooth execution, and are therefore best executed on-board the robots.

3. Vision Processing for Event Recognition

Important objects in the RoboCup environment are color-coded: the ball is colored orange, the field is green, and the two goals are colored blue and yellow. There are also color-coded markers intended for robot localization; each of these markers consists of two colored bands on top of a white column. To detect objects in the RoboCup environment, we perform color segmentation on the camera images using the CMVision⁹ image library and then run a series of object detectors over the segmented images to determine which objects are present.

We have implemented effective color-based object detectors for the ball, goals, and localization markers. Below, we discuss the purpose of vision on the robot, including ball detection as performed in CMCast. We also present a SIFT-based algorithm that enables the needed detection of multiple color-equivalent robots. The algorithm was not part of the demonstrated CMCast, but it has been successfully applied to RoboCup video footage.

3.1. Color-Based Object Detection for CMCast

We use vision to provide low latency information to the onboard behaviors so the robot can quickly and realistically respond to events as they unfold. In the commentator domain, the QRIOS mainly use low latency information from vision for

effective ball tracking. The robots use ball position estimates from the vision module to servo their heads and bodies to make it clear to observers that they are paying attention to the current state of the soccer game. In addition to tracking the ball, the robots use information from vision to detect the positions of localization beacons. Beacon information allows the robots to compute their global positions on the field and therefore translate the positions of other objects, such as the ball, into global coordinates, so that those positions can be shared between robots. Furthermore, the robots use vision of the goals to focus on the appropriate goal in response to events such as points being scored.

We use vision 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. Each robot uses its estimate of the ball position as well as its own position to detect when the ball enters particular regions of the field, such as the area around a goal. The robots also monitor the ball to determine when it is kicked. Each robot maintains a running history of ball positions. They compare movement vectors between the ball positions and compute a heuristic confidence metric, which gauges whether or not the ball has been kicked, based on the length of the vectors as well as the co-linearity of the vectors. In practice, a history length of six ball sightings, or one half seconds of data, allows the robots to accurately detect when the ball is kicked.

Object detection is performed after color segmentation. Figure 4(a) shows an example of a color image as captured by a robot's camera. Figure 4(b) shows the results of color segmentation on the same image. These images show the successful segmentation of the orange ball from the player robot and the field. A localization marker is also segmented and can be seen at the edge of the field.

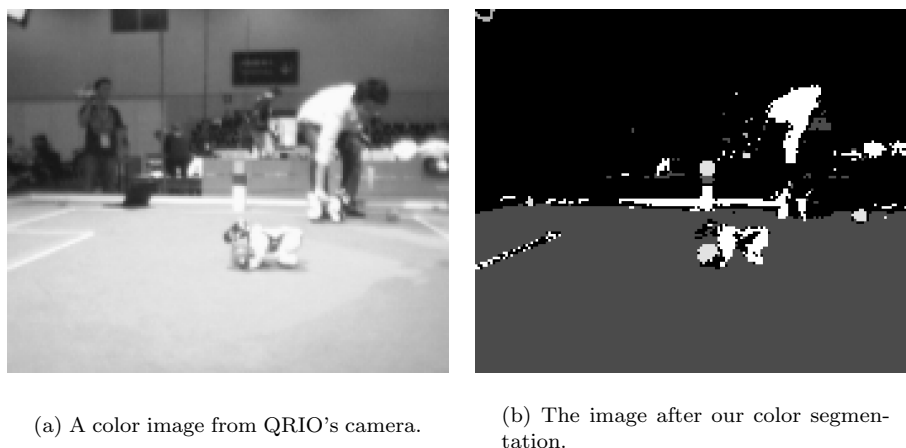


Fig. 4. Color-based image processing for recognition of relevant game objects.

The result of color segmentation is a set of colored regions. We run a series of object detectors over this set of regions. Each object detector selects candidate regions that are likely to belong to that object. For example, the ball detector considers all orange regions above a size threshold; the localization marker detector looks for an appropriate pair of colored regions next to each other (e.g. a pink region directly above a blue region). These candidate regions are scored according to a series of heuristic models. For instance, the models for the ball expect the ball region to have a roughly square bounding box and to be near green regions (corresponding to the field). Color regions possessing most of these features therefore score highly by the ball-detection model. The output of the ball detector is the largest orange region with an acceptably high score. This score is also a heuristic confidence estimate indicating how likely it is that the object is truly present. The distance to each object is calculated by intersecting rays through the closest pixels of the object onto the ground plane. The relative position of each object can then be found via the robot's kinematics; this relative position is then translated into a shared global coordinate frame by using the results of the Localization module.

3.2. SIFT-Based Detection of Player Robots

Our actually demonstrated humanoid commentators at RoboCup 2006 relied on visual perception of the ball and of the goals. However, we want to provide the humanoids with the additional capability of autonomously processing their own vision to evaluate a more complex game's state, in particular by determining the positions of the players on the field. We have successfully researched on such an algorithm, which we now present. We also show very promising experimental results obtained in the RoboCup domain.

The detection and position identification of multiple non-rigid objects, such as walking 4-legged robots, is a very different challenge from the problem of detecting single fixed-shape objects such as the ball in a robot soccer game. While the ball is a moving target that gets frequently occluded distorting its visible shape, it is still identifiable through its unique orange color on the field. The player robots however, are basically all of the same color and are very likely to share colors with the field markings and other objects in the background. In addition to this color ambiguity, we are interested in separating the multiple robots. This separation would be difficult to reliably achieve using color segmentation, as there is the risk of falsely identifying a cluster of robots as a single object due to an incorrect merging of neighboring colored regions.

As we cannot rely on color for the recognition and separation of the multiple robots, we introduce a robot detection algorithm to make use of visually "interesting" textural features of the object to be recognized. However, even when relying on textural features rather than color, we still have to overcome the challenge that highly non-rigid player robots cannot be easily represented by a single global feature descriptor because a robot's overall shape configuration can change fundamentally

between observations. Our approach aims to overcome this problem by representing objects using purely local features without enforcing a strict geometric relationship between them.

For the effective separation of the multiple robots, we introduce a probabilistic voting scheme where each detected local feature produces its own hypothesis of an object’s most likely location. Clustering then enables the detection of multiple instances of the robot in the image. Our algorithm provides detection of multiple robots in a scene while being robust enough to handle object deformation, object motion, and perspective changes which are inevitable in our commentator task.

Our algorithm’s feature detection and description is based on PCA-SIFT.¹⁰ A standard SIFT vector is a 128-dimensional local feature descriptor of interesting keypoints in the image.¹¹ SIFT descriptors are invariant to scale and rotation and relatively robust to perspective changes, making them ideal for object recognition tasks. SIFT has been successfully applied to robust object detection in still images¹¹ for a variety of applications, such as metric robot localization¹² and medical imaging¹³. A comparative study shows that SIFT descriptors are one of the best currently available descriptors.¹⁴ PCA-SIFT¹⁰ is an extension to SIFT that reduces SIFT’s high dimensionality by applying principal component analysis (PCA). PCA-SIFT is hence better suited for nearest neighbor lookups against a training dataset. We have used a 20-dimensional PCA-SIFT for the detection of multiple static objects in continuous video footage, a task that is commonly encountered in many robotic domains.¹⁵ We present an extension of this technique for domains containing multiple non-rigid objects.¹⁶

In order to detect the multiple walking similar robots, our approach consists of two major components: the *training stage* to learn the representation of a single player robot by collecting its PCA-SIFT features; and, the *recognition stage* to detect and localize player robots in real-time video footage.

3.2.1. Training for Learning an Object Description

The training stage processes a training video V containing one training object (e.g., the player robot), spanning a continuous spectrum of perspectives and motions, with the goal to build a library of the object’s local PCA-SIFT features. Table 1 shows the algorithm for the training stage.

The annotation mask acts as a way to single out the training object from a training video with background noise and other objects, such as a typical robot player in the RoboCup game footage, as observed by side-line robot commentators. To create the annotation mask, we developed an effective tool,^{15,16} where users annotate the object to be learned by drawing an approximate outline around it in the training video. The center c_i of the object is inherently annotated by calculating the center of mass of the object’s outline. The relative location $loc_{rel_{ij}}$ of each PCA-SIFT feature toward the center of the object c_i is furthermore computed as described in Table 1. All PCA-SIFT features lying outside of any annotated region

-
- (1) For each incoming frame v_i , treated as an independent observation,
 - (a) Generate the set of all PCA-SIFT *keypoints* K_i , such that keypoints $k_{ij} \in K_i$.
 - (b) Manually create an *annotation mask* m_i approximating the boundary and *center* c_i of the training object.
 - (c) Reject any keypoints from K_i lying outside of the *annotation mask* m_i .
 - (d) Store the relative location $locrel_{ij}$ of each keypoint towards the annotated object's center c_i , such that $locrel_{ij} = c_i - \text{loc}(k_{ij})$.
 - (2) Combine all retained *keypoints* k_{ij} into a single set T' .
 - (3) Generate the final training set T by reducing T' using agglomerative clustering.
-

Table 1. Training phase for the acquisition of a SIFT-based robot model.

are rejected from the training set.

Note that even after rejecting features that do not belong to the training object, there is still a very large dataset of relevant SIFT features. Considering that this data comes from a continuous video sequence, it can be assumed that many of the collected features are highly similar as there typically exists little variation between two frames. We take advantage of this redundancy and use an agglomerative clustering algorithm¹⁷ to further reduce the number of features in our dataset. The algorithm uses an Euclidean distance threshold in feature space to determine whether any two features should be merged into a cluster. When merging, we use the mean to compute the location of the newly merged feature. This methodology creates a controllable risk of occasionally falsely merging two descriptors.

At the end of the training phase, a training dataset is returned that corresponds to a descriptor of the object of interest for recognition.

3.2.2. Centroid Voting Space for Multi-Object Recognition

Table 2 shows the detection and position identification stage allowing for the recognition of multiple similar objects.

The detection threshold θ determines the matches with the training data, as the Euclidean distance in PCA-SIFT space between a feature and its nearest neighbor from the training dataset. Choosing a good value of θ is critical. A smaller value of θ will deliver fewer but more precise matches, while a larger value of θ will deliver more matches while increasing the likelihood of false positives.

We introduce a voting scheme to determine the position of each object (e.g., player robot) in the image. Any matched feature k_{ij} votes for where it predicts the center of its parental object to be. The algorithm retrieves the nearest neighbor t_l and its relative position towards the annotated object's center from the training set T . The algorithm then rotates and scales t_l 's relative location vector to match the scale and orientation of the newly detected keypoint k_{ij} . When adding this vector

Let the “voting space” be a set S containing votes of hypothesized 2D robot positions. Let $S = \emptyset$ initially.

For each frame v_i of our incoming continuous video V :

- (1) Generate the set of all PCA-SIFT keypoints K_i containing keypoints $k_{ij} \in K_i$.
- (2) For each keypoint k_{ij} of K_i :
 - (a) Perform a nearest neighbor lookup with all the elements t_l of the training set T . An observed PCA-SIFT feature k_{ij} is considered a match if

$$\min_{t_l \in T} D(k_{ij}, t_l) \leq \theta$$

where D is the Euclidean distance, and θ is a given detection threshold.

- (b) For any matching k_{ij} , calculate the hypothesized position of the object’s center p_l , by scaling and rotating the previously recorded relative position $locrel_l$ of t_l to match the scale and orientation of k_{ij} . Add the hypothesized position p_l as a “vote” to the voting space S .
- (3) Run a clustering algorithm on the voting space S using a clustering threshold δ and enforcing a minimum cluster size of s votes. Then calculate the center of mass for each cluster, as the resulting object position in the image.

Table 2. Detection and position identification phase of the SIFT-based multi-robot detection.

to the absolute location of k_{ij} , an hypothesis of the object’s center is generated. This hypothesis is counted as a vote for the object’s center in the two-dimensional voting space. This voting scheme allows for the identification of multiple hypotheses for the multiple object centers.

The voting scheme is an heuristic approximation, as there is no clear guarantee that the same feature cannot occur at a different relative position towards the object’s center than in the training data. This is especially the case because the object is non-rigid. However, we can optimistically assume that many of the votes do in fact approximately match the object’s center and that our voting scheme takes care of possible outliers.

An interesting side-effect of the voting space approach is that it automatically solves the problem of occlusion. Even when an object is half occluded or moving off the visible screen, the algorithm still correctly hypothesizes the object’s center.

When applied to multiple objects in the image, the algorithm finds multiple density peaks of votes in our voting space, each representing a hypothesis for a player robot’s location. Thus, after populating the voting space, a clustering algorithm locates these peaks. We use the Mean-Shift¹⁸ clustering algorithm due to its high performance even under large sets of votes. As other clustering techniques, Mean-Shift requires a clustering threshold δ , also known as the “bandwidth”. The crucial purpose of δ is to determine whether a particular vote is close enough in feature

space to be considered a part of an already existing cluster of votes, or whether it is too far and should thus be treated as the initial vote of a new cluster. Choosing a smaller value of δ makes the clustering of votes less likely, and can lead to unnecessary multiple detections of the same object. Choosing a larger value of δ increases the likelihood of clustering votes and increases the risk of wrongly grouping multiple object instances into a single detection point. Finally, clusters are rejected, if they contain less votes than a minimum cluster size threshold s . The center of mass is computed for each remaining cluster and returned as the final detected position.

3.2.3. Multi-Robot Detection Results

The domain of our commentating task certainly constitutes an interesting testing ground from a vision perspective. The robots used are highly dynamic and non-rigid, featuring 20 degrees of freedom that allow almost any conceivable actuation of legs, feet, neck, and head. Participating RoboCup teams typically create their own unique robotic motions which range from basic walking patterns to rather complex and unorthodox bodily expressions. Video scenes are normally populated with several robots that can be depicted in various shape configurations, perspectives, and scales. Other frequent vision constraints are robot occlusion, highly cluttered backgrounds (for instance by a human audience), and motion blurring. We applied our approach on random training and testing data from this domain.

Figure 5 shows some interesting detection results using real RoboCup footage.

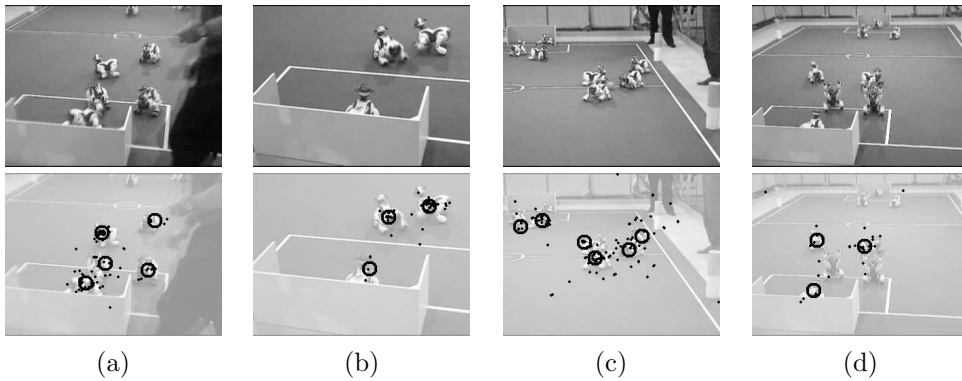


Fig. 5. A selection of interesting frames from the testing video. The top row shows the input frames. The bottom row shows the centroid voting space after running PCA-SIFT on the input frames. Each vote is annotated by a small black dot. The result of clustering and centroid-finding on this voting space is marked by larger black circles.

Frames (a)-(c) show some very successful examples where the algorithm is able to correctly locate the robots even when being partially occluded, such as the goalie in frames (a) and (b). Frame (c) furthermore shows how the algorithm is able to correctly recognize robots of different scale and orientation, even when clustered

closely together. Finally, frame (d) shows an example where two robots in the front are performing a rare “celebration” move. Detection fails for these two robots, because none of their visible local features were ever observed during training. Nevertheless, the other robots in the front of this scene are successfully detected.

Good computational performance is a significant requirement of our approach as it is intended to be used in a real-time commentator task. We have performed a detailed quantitative analysis of our algorithm that shows its effectiveness.¹⁶

4. Wireless Information Sources

Wireless networking allows each of the components of our architecture, including the QRIO robots, to communicate with each other.

4.1. Game Controller

The referees of the RoboCup four-legged robot league use a software program called the Game Controller, which wirelessly sends referee calls and similar information to the players. Figure 6 shows the Game Controller interface.



Fig. 6. Screenshot of the Game Controller interface, as developed in the RoboCup Four-Legged Robot League.

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 the team 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 which robot(s) were penalized for each foul.

- Goal scored
- “Ball out” calls made by the referees.
- Time-outs called by either team.

The Game Controller provides valuable notification of many events that the robots may otherwise have been unable to recognize autonomously.

4.2. *Puppet Master*

There are many significant number of interesting events that are not detectable either with the on-board vision or with the Game Controller input. For instance, the AIBOs sometimes crash due to empty batteries or software errors. It is difficult for the commentators to visually recognize a crashed robot (as opposed to a stopped robot). For situations like these, we have developed the off-board Puppet Master controller to allow a human operator to artificially insert specific types of events, such as robot crashes, into the commentators’ game history. Figure 7 shows a screenshot of the Puppet Master interface.

The Puppet Master consists of three main functional parts, namely:

- **Event guidance** - One of the main functions of the Puppet Master is to provide the robot commentators with additional event information. The top of the interface in Figure 7 shows the event choices available to the user. The events fall into three categories:
 - *Game events* are predefined announcements associated with game situations. These events include sequences of announcements made at various stages of the match (before the game, at half time, etc.), exclamations relating to interesting occurrences on the field (“wow!”, “oops!”, etc.) and comments on common game occurrences (robot crashed, nice save, etc.).
 - *Filler events* are used to fill in the silence when nothing particularly interesting is happening in the game. 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.
 - *Manually selected speech and motion events* enable the user to specify custom speech and gestures which are not part of the predefined events (shown under the “Say”, “Motion” empty fields at the top of the interface). The human can hence enable a robot to respond to unpredicted and highly unusual events. Within the demonstrated CMCast at the RoboCup 2006 event, this capability was not required during commentating of the multiple games.



Fig. 7. Screenshot of the Puppet Master – an interface for human guidance.

- Perceptual guidance** - To avoid interfering with the human referees moving around the field, the CMCast robot commentators remain in fixed positions on the side line of the field (see Figure 1). Commentator movement is restricted to rotating in place in order to face the area of interest. As a result of these limitations on movement, the commentators are not able to track the ball continuously due to occlusions in distant areas of the field. Additionally, the ball position often changes suddenly when the ball is manually positioned by the human referee following a ball-out event. The perceptual guidance feature of the Puppet Master enables the user to provide the commentators with the ball position. The input is given at a high-level of granularity, as we discretize the field into 25 cells, as shown in Figure 7, with the two blue (B) and yellow (Y) goals. Based on this information, the robot rotates to face the designated area and continues tracking the ball position using the on-board vision.
- Motion guidance** - Finally, the announcing motions may cause the robots to shift from their desired positions next to the sideline. The Puppet Master enables each robot to be remotely repositioned using the arrow interface located on the bottom of the screen. Remote positioning through the Puppet Master is

used instead of automatic localization in order to ensure that noise and odometry errors do not cause the robot to step into the field and interfere with normal game play.

In summary, the Puppet Master provides a sliding autonomy interface for integrating possible human guidance within the robot autonomous control. As autonomous detection of game events improves, the need for this level of control decreases.

4.3. *Commentators*

The final source for event detection are the commentator robots themselves. As described in the previous section, each robot uses its on-board vision to track the location of the ball. Based on this information, each robot provides the Director with two types of information:

- Ball location on the field
- Kick event description (direction and velocity)

In addition to information obtained from vision, the robots report back to the Director on the execution of their motion commands. This practice is necessary for tighter synchronization, since motion commands can take up to several seconds to complete. The Director then synchronizes the two robots' motions, as well as each individual robot's gestures with its speech. In the current implementation of CMCast, speech is generated by an off-board text-to-speech component¹⁹ and sent to external loudspeakers. The external sound generation and amplification is needed so that the robots' speech can be easily heard in the noisy stadium environment. However, the robots' on-board speech synthesis software can be easily used instead.

5. Commentating Action Selection: Speech and Gesture

The purpose of the QRIO commentator team is to inform and entertain. As such, the commentator algorithm is tasked with intelligently processing vision and other inputs in order to generate a timely and relevant commentary that coordinates the two robots' speech and gestures. The commentary is largely event-driven, but also remarks on the development of the game over time. In this final section, we provide details of the commentator algorithm, located centrally in the Director module, that processes its inputs (robot vision, Game Controller events, and Puppet Master directives) to generate commentary (speech and gestures).

Formally, the task of the commentator algorithm is to map observations to actions. We break up this task into three subtasks: 1) keep a game history of sufficient statistics of the raw observations; 2) detect relevant events from the history (in many cases, a relevant event is immediate and only a function of the current observations); and 3) select an action to commentate the event. Therefore, we break up this section into the subsections of game history, event detection, and action selection.

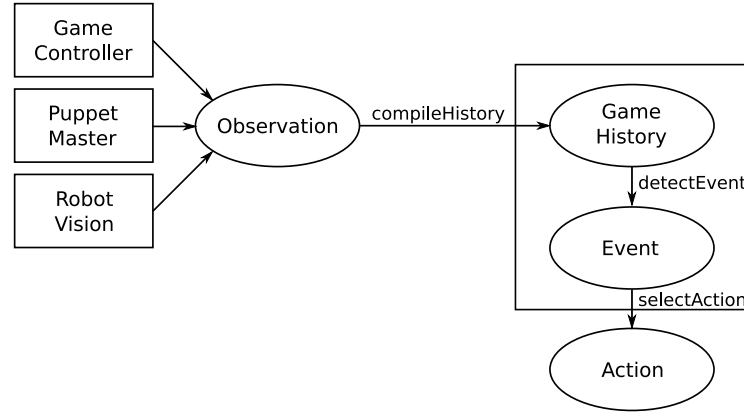


Fig. 8. The internal structure of the Director module of the CMCast global architecture (Figure 2).

Figure 8 depicts the flow of data of the internal commentating algorithm within the Director, supporting our detailed presentation on the three subtasks.

5.1. *Game History*

One challenge the commentator algorithm faces is to filter the raw inputs (as described in Section 2) and summarize them in order to act. The full history would be the sequence of all observations at all timesteps; however, the full history is large and unwieldy for use in defining a policy mapping histories to actions. We only wish to consider elements (or features) of the history that may be relevant to later commentary. Hence, we introduce a *Game History* $s \in \mathcal{S}$, where s is a tuple of variables each of which corresponds to a single game statistic. The Director collects and summarizes its inputs into the *Game History* s .

We denote this summarization process as the *history-compilation function*, denoted as $\Theta : \mathcal{O} \mapsto \mathcal{S}$. An example statistic is defined as $\Delta o = o_t - o_{t-1}$, which allows the algorithm to detect goals, since the output of the Game Controller includes only the score at each timestep. A more complex statistic is the current “streak” for the most recently scoring team (i.e., how many goals it has scored without answer). Finally, the trivial “identity” statistic is of note, which maintains the most recent observation o_t .

In general, maintaining this sort of history allows comments relating to the development of the game over time; formally, it transforms the non-Markovian process over instantaneous game state information $o \in \mathcal{O}$ into a Markovian process over the Game History variables $s \in \mathcal{S}$, where the Game History is a sufficient statistic for the commentator team to act. This sufficiency property is why we denote the game history with the letter s , since it is essentially the *state* of the game, from the point of view of the commentator algorithm.

5.2. Prioritized Rule-Based Event Identification

Now that we have filtered our observations to only consider relevant information, we must determine when to act. Vision processes information many times per second, but it is certainly not necessary to change actions at that rate; although motion control might still operate at high frequency, high-level decisions about how to commentate the game should not.

We consider *events* $e \in \mathcal{E}$ as those changes in game state that are “significant,” as we define below through the use of a set of predefined rules or predicates. Many events are determined by $\Delta o = o_t - o_{t-1}$; these correspond to announcing immediate changes in the game state, such as a goal (a change in the game score). However, more generally, events such as the “streak” mentioned in the previous subsection require use of the game history s .

We define the *event-identification function* as $\mathcal{P} : \mathcal{S} \mapsto \mathcal{E}$ to capture a significant event from the game history. Since the set of possible game statistics is quite large, we specify the event-identification function using an ordered set of *rules*, or *predicates* (p_1, p_2, \dots, p_n) that apply to the Game History s . Each predicate p_i has an associated event and is a boolean-valued function. When $p_i(s)$ evaluates to either true, the associated event is detected. The order of the n predicates in the predicate list indicates priority; we define the function \mathcal{P} in terms of the predicates as follows:

$$\begin{aligned} \mathcal{P}(s) = e_i \text{ if:} \\ & p_i(s) \text{ is true, and } p_j(s) \text{ is false } \forall j < i \\ \mathcal{P}(s) = \text{“null” if:} \\ & p_i(s) \text{ is false } \forall i \end{aligned}$$

where “null” is the special empty event for the case when all predicates are false.

The behavior of the algorithm is that the list of predicates is examined, one after another according to their order, and if a predicate p_i evaluates to true, it “fires” its associated event e_i and no more predicates are processed. The ordering of the predicates captures the relative significance among events. For example, goals have higher priority than other events, as they need to be promptly announced. Although many predicates might be true (i.e., many events might have occurred simultaneously), only a single one, the most significant, is announced.

Additionally, the Puppet Master can generate events directly. This does not break the formalism above, since we define a predicate and a corresponding event for each Puppet Master signal. These predicates are listed first, so Puppet Master commands have the highest priority; conceptually, if a human issues a command through the Puppet Master, the event is passed straight through to become a detected event.

5.3. Action Selection

The commentator algorithm uses the events $e \in \mathcal{E}$ to create a basic output as an *action* $a \in \mathcal{A}$. An action consists of (one or more) spoken phrases $u \in U$ and gestures $g \in G$. An action may be performed by one or both QRIOs, and may be a single phrase/gesture pair or a sequence of several phrases and gestures. When an action uses a sequence with both QRIOs, in a turn-taking fashion, a fully-coordinated dialog is achieved. In particular, we use this dialog at the beginning of the game for the QRIOs to make initial comments on the game, including announcing facts about the teams in the game and predictions on which team the robots think will win, similarly to many human sports commentators.

Formally, the *commentator function* $\mathcal{C} : \mathcal{E} \mapsto \mathcal{A}$ maps events $e \in \mathcal{E}$ to actions $a \in \mathcal{A}$. We construct \mathcal{C} through the following steps:

- (1) An *atomic action* of a robot r at time n is $a_n^r \in A$. It consists of both an utterance and a gesture, represented as an ordered pair (u, g) .
- (2) A *joint action* of both robots at time n is an ordered pair of atomic actions $(a_n^1, a_n^2) \in A \times A$.^c
- (3) An *action* is a sequence of joint actions of the two robots over time: $\mathcal{A} = ((a_1^1, a_1^2), (a_2^1, a_2^2), \dots, (a_n^1, a_n^2), \dots) = (A \times A)^*$. Different actions have different sequence lengths.
- (4) The commentator function is the composition of a deterministic function \mathcal{C}_1 mapping events to sets of possible actions and a non-deterministic function \mathcal{C}_2 for choosing a single action from this set:

$$\mathcal{C}_1 : \mathcal{E} \mapsto 2^{\mathcal{A}} \text{ and } \mathcal{C}_2 : \text{range}(\mathcal{C}_1) \mapsto \mathcal{A}, \text{ such that } \mathcal{C} = \mathcal{C}_2 \circ \mathcal{C}_1.$$

The importance of \mathcal{C}_1 is that it generates predictable robot behavior; for example, when a goal occurs, the robots need to announce the goal. However, we do not want them to announce it the same way every time, nor do we want to prescribe how to announce it differently the first time, and any other time. Hence, we introduce \mathcal{C}_2 , which allows the robots to probabilistically and autonomously select from a varied set of actions. For example, the actions for a goal event range from the simple sentence “Goal!” to the more elaborate “An astonishing goal for the Blue team!” followed by a celebratory dance. We specify \mathcal{C}_1 via an *Action Library*, consisting of a list of potential actions for each event. For each event, \mathcal{C}_1 returns a set of actions (in the power set of \mathcal{A}), out of which one action is then returned by \mathcal{C}_2 .

We specify \mathcal{C}_2 by maintaining a weight for each action w^a , and the algorithm samples the final action a to be returned from the list of actions returned by \mathcal{C}_1 with probability proportional to w^a .^d

^cIn general, we could use R robots, such that joint actions would be in A^R .

^dWe could instead choose $a = \arg \max_w$, but we further simulate varied “replayability,” by choosing the action through a non-deterministic sampling, even if we update the weights with time.

To prevent repeating an action too much, we reduce the weight by a fixed quantity Δw^a every time the action is used. Furthermore, we wish to prevent an action from being used in quick succession. As such, we introduce a “cool-down” effect by modifying the actions’s weight over the time since its last use t such that: 1) the probability of it being selected again immediately is zero, and 2) as $t \rightarrow \infty$, its probability should return to its base weight. The equation we use to achieve these properties is:

$$w^a(t) = (w_0^a - \alpha_t^a \Delta w^a) \exp\left(-\frac{1}{t}\right)$$

where w_0^a is the initial weight of action a and α_t^a keeps a count on the number of times the action has been visited.

Note that we use different values of w_0^a and Δw^a for different actions, since some phrases might be less pleasing or “get old” faster than others. Also note one important consequence of our scheme: the maximum number of times an action can be used is $\frac{w_0^a}{\Delta w^a}$, as the weight becomes zero when $\alpha_t^a = \frac{w_0^a}{\Delta w^a}$. After this point, the action will never again be selected. We can set w_0^a and Δw^a with values that control the amount of use of each action. If $\Delta w^a = 0$, then the action is always selected probabilistically with its initial weight w_0^a .

By defining an action as a sequence of speech/gesture pairs for each robot, we can achieve a wide variety of behaviors. In addition, we allow special “null” utterances and gestures, so that this sequence indeed captures behavior that is just speech or that uses only a single robot. For example, a full dialog action might be the sequence:

t=1: ((“Excellent Goal!” , LiftArms), (null, Cheer1))
t=2: ((null, null), (“The fourth goal for blue in this game!” , TurnToRobots))
t=3: ((“Yes, blue is doing very well!” , LightUpLEDs), (null, LightUpLEDs))

where LiftArms, TurnToRobots, Cheer1, and LightUpLEDs are predefined gestures.

Table 3 summarizes the complete commentator algorithm in pseudo-code, according to our description. The procedures are part of Director, as shown in Figure 8.

5.4. Examples of Events and Actions

A library captures the mapping from general (variabilized) events to actions of (variabilized) utterances and gestures. Events, as generated from the compiled Game History, are conceptually associated with an original source of input. Table 4 shows a very short subset of the events and actions in the library, as well as the source of the information used to generate the event, for clarity.

For example, the first event in the table is “Goal scored by ?Tx,” where ?Tx is a particular team. The source of this event is the Game Controller (GC). Three of many other actions the robots can choose to execute in response are shown in the table. In the first action, the robot Commentator 1 says “What a great goal by ?Tx,” while executing the gesture “LiftArms,” and the robot Commentator 2

procedure compileHistory(s, o):

1. $\Delta o = o_t - o_{t-1}$
2. otherStats = updateStats(s, o)
3. $s \leftarrow o \cup \Delta o \cup \text{otherStats}$

procedure detectEvent(s):

1. for p_i in predicates:
2. if $p_i(s)$:
3. return e_i
4. return null

procedure selectAction(event):

1. actions = ActionLibrary(event)
 2. weights = $w^a(t)$ for all $a \in \text{actions}$
 3. $a = \text{sampleAction}(\text{actions}, \text{weights})$
 4. α_t^a = number of times action a has been selected
 5. Δw^a = weight update for action a
 6. $w^a(t) \leftarrow (w_0^a - \alpha_t^a \Delta w^a) \exp\left(-\frac{1}{t}\right)$
-

Table 3. The commentator algorithm to select an action for a detected event, as part of Director

simply executes the gesture “Cheer 1.” We can see an example of the Commentators speaking in unison, (“Goal!”) while carrying out different gestures. Table 4 shows an example of an event that is originated from the Game History (GH), namely the “First goal” event. In this action example the robot Commentators conduct a conversation: “?Tx scores first in the game!” while executing gesture “LightUpLEDs”; “Can team ?Ty catch up?”, while executing gesture “Dance1.”

Through the use of a large library of utterances and gestures, and an algorithm that intelligently selects which events to act on, and which actions (utterances and gestures) to use, we are able to achieve commentary that is entertaining and informative. Our approach furthermore ensures that the robots’ commentary is timely and coordinated.

6. Conclusion

The commentator task is an interesting domain for humanoid robots. We present our work on the first two humanoid commentators for the RoboCup AIBO robot soccer games. The game is played on a large field, is highly dynamic, and is hard to completely specify an a priori 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 RoboCup 2006 event in multiple robot soccer AIBO games. The robot commentators successfully and autonomously ob-

Event	Src.	Robot Commentator 1		Robot Commentator 2	
		Utterance	Gesture	Utterance	Gesture
Goal scored by ?Tx	GC	What a great goal by ?Tx!	LiftArms	<i>null</i>	Cheer1
		<i>null</i>	Cheer2	Incredible goal!	Cheer3
		Goal!	Cheer2	Goal!!	LiftArms
First goal	GH	?Tx scores first in the game!	LightUpLEDs	<i>null</i>	Dance1
		<i>null</i>	Cheer2	Can team ?Ty catch up?	TalkGest1 TalkGest1
?N push robot penalty team ?Tx	GC	That was the ?Nth robot pushing foul.	TalkGest1	<i>null</i>	<i>null</i>
		<i>null</i>	<i>null</i>	?Tx has been called for robot pushing.	TalkGest3
Nice kick	RV	Nice kick Red Team!	Cheer1	Wow!	<i>null</i>
Nice save	PM	<i>null</i>	Cheer2	Nice save!	Cheer3
		Good defense!	Cheer2	<i>null</i>	Cheer3
Power play team ?Tx	GC	Many ?Tx robots are out of the field. Great chance for ?Ty.	TalkGest2	<i>null</i>	<i>null</i>
		<i>null</i>	<i>null</i>	Yes! Can ?Ty take advantage of it?	HeadNod1
1mn left	GC	One minute in the game!	TalkGest2	Go teams!	TalkGest3

Table 4. Very short example of the library of mapping of CMCast events to actions and their detection sources (Src.), including Game Controller (GC), Puppet Master (PM), robot vision (RV) and Game History (GH). Events and utterances include variables (marked with prefix “?”) instantiated in the game, for the team name, team score, and event counters.

served the game and announced the events through varied utterances and motion adapting to the sequence of the game and the different teams.

Acknowledgments

We thank SONY for making the QRIOs available for our research, enabling our development of this particular interesting commentator task. We further thank SONY for their multiple developed motion and software features, which underlie our developed CMCast system. Finally, we thank the organizing committees of RoboCup 2006 and of the RoboCup Four-Legged Robot League for allowing the demonstration of CMCast at the International RoboCup 2006 in Bremen, Germany.

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.
10. Y. Ke and R. Sukthankar, "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors," *Proc. CVPR*, vol. 2, pp. 506–513, 2004.
11. D. Lowe, "Object recognition from local scale-invariant features," *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, pp. 1150–1157, 1999.
12. S. Se, D. Lowe, and J. Little, "Vision-based mobile robot localization and mapping using scale-invariant features," *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 2, 2001.
13. M. Moradi, P. Abolmaesoumi, and P. Mousavi, "Deformable Registration Using Scale Space Keypoints," *Proceedings of SPIE*, vol. 6144, p. 61442G, 2006.
14. K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 257–264, 2003.

15. S. Zickler and M. Veloso, "Detection and Localization of Multiple Objects," in *Proceedings of Humanoids 2006*, Genoa, Italy, December 2006.
16. S. Zickler and A. Efros, "Detection of Multiple Deformable Objects using PCA-SIFT," in *Proceedings of AAAI 2007*, Vancouver, July 2007.
17. K. Gowda and G. Krishna, "Agglomerative clustering using the concept of mutual nearest neighborhood," *Pattern Recognition*, vol. 10, no. 2, pp. 105–112, 1978.
18. Y. Cheng, "Mean Shift, Mode Seeking, and Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, 1995.
19. "Cepstral Text-to-Speech," <http://www.cepstral.com/>



Manuela M. Veloso is Herbert A. Simon Professor of Computer Science at Carnegie Mellon University. She received a *licenciatura* in Electrical Engineering in 1980, and an M.Sc. in Electrical and Computer Engineering in 1984 from the Instituto Superior Técnico in Lisbon. She earned her Ph.D. in Computer Science from Carnegie Mellon in 1992. Veloso researches in planning, control learning, and execution algorithms, in particular for multi-robot teams.

With her students, Veloso has developed teams of robot soccer agents, which have been RoboCup world champions several times. She is a Fellow of AAAI, the Association for the Advancement of Artificial Intelligence, an IEEE Senior member, and the President Elect (2008) of the International RoboCup Federation.



Nicholas Armstrong-Crews is a Ph.D. student in the Robotics Institute at Carnegie Mellon University. He hails from Alaska, where he studied computer science, mathematics, and natural sciences at the University of Alaska Anchorage. He is co-advised by Manuela Veloso and Geoffrey Gordon, with whom he studies decision theory under uncertainty for application in robotics.



Sonia Chernova is a Ph.D. student in the Computer Science Department at Carnegie Mellon University. She received her B.S. in computer science and robotics from Carnegie Mellon University in 2003. Her research interests include learning in robotic systems and human-robot interaction.



Elisabeth Crawford is a Ph.D. student in the Computer Science Department at Carnegie Mellon University. She has a BSc. Hons. (I) from the University of Sydney, Australia. Her research interests include multiagent learning and negotiation.



Colin McMillen is a Ph.D. student in the Computer Science Department at Carnegie Mellon University. He has worked in the RoboCup Four-Legged Robot League since 2003, focusing on high-level team strategy and multi-robot coordination. In particular, his dissertation aims to address how agents in timed, zero-sum games should change strategies based on the score of the game and the time remaining.



Maayan Roth received her Ph.D. from the Robotics Institute at Carnegie Mellon University in 2007. Her thesis, “Execution-time Communication Decisions for Coordination of Multi-agent Teams,” explores the use of communication heuristics to improve the performance and tractability of decentralized partially observable Markov decision processes (Dec-POMDPs). Currently, Maayan is working as a software engineer at Google in Haifa, Israel.



Douglas Vail is a Ph.D. student in the Computer Science Department at Carnegie Mellon University. He has worked with the AIBO robots and RoboCup since 2002. His research interests include robotics, machine learning, and activity recognition. In particular, he is interested in feature selection in conditional random fields for activity recognition in robot domains.



Stefan Zickler is a Ph.D. student in Computer Science at Carnegie Mellon University. He received his B.A. in Cognitive Science with minors in Computer Science and Linguistics from SUNY Buffalo in 2005. His main research focuses on physics-based behavioral motion planning for robot applications. Other research interests include real-time computer vision and multi-agent behavioral control.