IEICE TRANS. INF. & SYST., VOL.E92–D, NO.9 SEPTEMBER 2009

PAPER

Development of an Interactive Augmented Environment and Its Application to Autonomous Learning for Quadruped Robots

Hayato KOBAYASHI[†a)], ***Student Member***, **Tsugutoyo OSAKI**[†*], **Tetsuro OKUYAMA**[†**], **Joshua GRAMM**[††], **Akira ISHINO**[†***], ***and*** **Ayumi SHINOHARA**[†], ***Nonmembers***

SUMMARY This paper describes an interactive experimental environment for autonomous soccer robots, which is a soccer field augmented by utilizing camera input and projector output. This environment, in a sense, plays an intermediate role between simulated environments and real environments. We can simulate some parts of real environments, e.g., real objects such as robots or a ball, and reflect simulated data into the real environments, e.g., to visualize the positions on the field, so as to create a situation that allows easy debugging of robot programs. The significant point compared with analogous work is that virtual objects are touchable in this system owing to projectors. We also show the portable version of our system that does not require ceiling cameras. As an application in the augmented environment, we address the learning of goalie strategies on real quadruped robots in penalty kicks. We make our robots utilize virtual balls in order to perform only quadruped locomotion in real environments, which is quite difficult to simulate accurately. Our robots autonomously learn and acquire more beneficial strategies without human intervention in our augmented environment than those in a fully simulated environment.

key words: *augmented reality, autonomous learning, four-legged robot, robocup*

1. Introduction

The experiments on real robots, especially quadruped robots, take much more time and cost than those on PCs. It is also an enormous difference since we often need to consume a lot of energy for treating physical objects such as robots. The use of virtual robots is one of the efficient methods to avoid those difficulties, and so there are many studies on dynamic simulated environments [1]–[7]. However, since simulated environments cannot produce complete, real environments, we finally need to conduct experiments in the real environments where basic skills heavily depend on complex physical interactions.

In this paper, we propose a system for an interactive *augmented environment*, which serves as a bridge between simulated environments and real environments. Augmented environments, also known as *Augmented Reality* (AR), involve the real environments combined with some data generated by computers. By using our system, robots can obtain both the precise positions of physical objects and the imaginary positions of virtual objects despite being in a real environment. Human programmers can get debugging information overlaid on a soccer field instead of that in a console terminal, allowing them to concentrate on observing the behavior of the robots in the field. Moreover, both robots and human programmers can interactively touch and move virtual objects. Our augmented environment, in a sense, plays an intermediate role between simulated environments and real environments; considering the fact that we can simulate some parts of real environments.

Our system is another implementation of the system of Stilman et al. [8] and promotes their study. They proposed the novel paradigm for robot experimentation that takes advantage of augmented reality to enable unified testing of individual subsystems. The main difference about implementation is that we utilize camera input and projector output, while they utilized motion capture input and display output. Projector output is more intuitive than display output, since some useful data are directly projected on real environments. Actually, several studies [9], [10] adopted projector output and developed intuitive augmented environments for robots, although their motivation is different from ours. The significant point of this paper is that our system provides us with touchability of virtual objects owing to projectors, as well as perceptibility of them.

This paper focuses on the RoboCup Standard Platform League (SPL) [11] as a target application. RoboCup [12] is a competition for autonomous robots that play soccer and is an interesting and challenging research domain because it has a noisy, incomplete, real-time, multi-agent environment. SPL is one of the leagues in RoboCup, where all teams compete with identical robots permitted by the regulation, i.e., the Sony AIBO robots and the Aldebaran Robotics humanoid Nao robots. Since it is quite difficult to simulate legged movements completely, SPL should be one of the applications where our system can make great contributions.

The study of Guerra et al. [13] is closely related to our study, since they also focus on RoboCup soccer. They proposed the CITIZEN Eco-Be! league (recently, also known as the mixed reality competitions in the simulation league) where CITIZEN's mini robot Eco-Be! plays soccer in an augmented environment created by a display and a camera. This league gave new interesting and challenging research issues in RoboCup. Our augmented environment can be regarded as a generalization of the technique in the Eco-Be!

Manuscript received October 24, 2008.
Manuscript revised March 26, 2009.

[†]The authors are with the Graduate School of Information Sciences, Tohoku University, Sendai-shi, 980–8579 Japan.
[††]The author is with University of California, San Diego, U.S.A.
[*]Presently, with Toshiba Corporation.
[**]Presently, with Panasonic Corporation.
[***]Presently, with Google Corporation.
a) E-mail: kobayashi@shino.ecei.tohoku.ac.jp

DOI: 10.1587/transinf.E92.D.1752

league to the other real robot leagues. We can easily apply our techniques to different types of robots that are used in other leagues, although we focus only on AIBO robots in this paper.

As an application in the augmented environment, we address the learning of goalie strategies in penalty kicks. The goalie strategies involve the skills to save an incoming ball kicked by an opponent player, which is critical to not lose the game. The learning can be also regarded as the two dimensional extension of the study of Kobayashi et al. [14]. They studied the *autonomous learning* to trap a moving ball by utilizing reinforcement learning. The goal of the learning was to acquire a good timing in initiating its trapping motion autonomously, depending on the distance and the speed of the ball, whose movement was restricted to one dimension. The two dimensional extension would need the troublesome programs to find a rebounded ball and return it to the initial position, as well as an accurate localization system. In contrast, our robot in augmented environments can avoid these difficulties by adopting a virtual ball that can be positioned and controlled arbitrarily, where the precise positions of the robot and the ball are measured and calculated by the system. The advantage of this method is that we can easily achieve autonomous learning without human intervention, despite performing quadruped locomotion in real environments, which heavily depends on complex physical interactions and is quite difficult to simulate accurately.

The remainder of this paper is organized as follows. In Sect. 2, we describe our system to create an augmented environment for autonomous soccer robots. In Sect. 2.4, we show the portable version of our system that does not require ceiling cameras. In Sect. 3, we propose an autonomous learning method of goalie strategies in penalty kicks and make our robot learn and acquire goalie strategies on their own. Finally, Section 4 presents our conclusions.

2. Augmented Soccer Field System

2.1 Overview of Our System

In this section, we describe the *Augmented Soccer Field System* for our augmented environment. This system consists of a pair of cameras and a pair of projectors in the ceiling with one computer to control them. The process in the computer is separated into two programs: a *recognition program* and a *virtual application*. The recognition program is a program that recognizes objects, such as robots and balls, in a soccer field based on the images from the ceiling cameras. The virtual application is a program that calculates real coordinates in the field based on the results of the recognition program and then sends the positions of the objects to the robots. It also detects collisions between the objects.

Figure 1 shows the overview of our whole system. Each pair of cameras and projectors was set up on the ceiling located in the center of each half of the field so as to

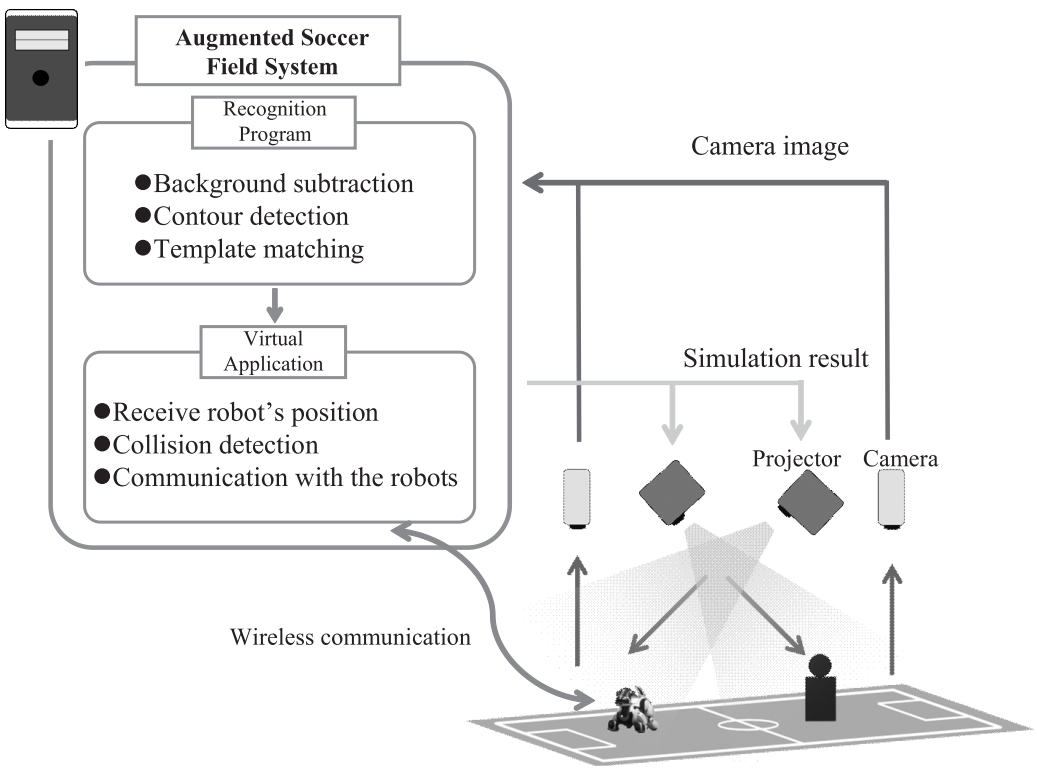


Fig. 1 Overview of our system, which is separated into two programs: a recognition program that recognizes objects with a camera and a virtual application that projects generated data with a projector.

cover the whole field, since one pair can take images with a camera and projects generated data with a projector for a half of the field. We use Dragonfly2 of Point Grey Research Inc. as a camera and LP-XL45 of Sanyo Electric Co., Ltd. as a projector. The projector is the only light source in the room when the system is running, that way humans can sufficiently recognize projected images on the field. Each projector projects the farther half of the field, since the distance from the ceiling to the field, which is about 2.5 meter height, is too small to cover the whole field.

2.2 Recognition Program

Camera images are captured into the computer by utilizing the API for Dragonfly2, and then passed to an object recognition process with the Open Source Computer Vision Library (OpenCV). The processes for two cameras are multithreaded, since they are independent of each other.

Captured camera images are quite distorted since our cameras have wide-angle lenses. The distortion of images can easily lead to some errors in the object recognition process for the images. Thus, we must estimate adjustment parameters for each camera so as to correct such distortion.

We also need to estimate different adjustment parameters (i.e., a projection matrix) for transforming the coordinates of recognized objects in camera images to those in the field. In our system, we obtain the projection matrix by referring 8 points that consist of 4 corners of the half field and 4 corners of the penalty area. Since our cameras and projectors are fixed in our room, it is sufficient to estimate those parameters only once unless their positions are changed.

2.2.1 Recognition Method

We utilize the *background subtraction* method [15] in order to recognize objects on the field. The reason comes from the fact that the object recognition based on colors is usually difficult because the colors can change depending on lighting conditions. The background subtraction method is designed to store a background image without any objects in advance, judge whether or not each pixel in a target image in process belongs in the background image, and separate the target image into a foreground image and the background image. In this paper, the background image means the image of our soccer field without any robots and balls.

We create a masked image that indicates the regions of the objects to be recognized by utilizing the background subtraction method and then identify each object by extracting the contour of the object. Since our robot has a direction (or angle), we identify the orientation of each robot by utilizing template matching.

2.2.2 Calculation of Background Subtraction

The most naive subtraction method is to calculate color subtraction between a current target image and the background image for every pixel and then judge whether the pixel belongs to the background image based on a certain threshold. In general, however, the criterion for each pixel should be changed depending on its location, since brightness varies with location. Thus, we suppose that the degree of color change for each pixel follows a normal distribution $N(\mu, \sigma^2)$. We calculate the mean μ and variance σ^2 of each pixel by capturing 100 background images and regard the pixel as background if the difference between μ and the color value of the pixel is greater than $c\sigma$ with some constant c. Each image has 3 color channels (RGB), and so we separately treat each channel, in the way that the constant c of a channel is independent of those of the others. We define that the pixels regarded as background must satisfy the above condition in every channel.

2.2.3 Extraction of Contour

We extract the contour of each object from a masked image calculated by the background subtraction method in order to identify the object. Since there can be unknown objects other than robots and balls in the masked image, we utilize the length of a contour line and the size of a contour area for eliminating unknown objects. Then we can obtain the minimum bounding box covering each identified contour as shown in Fig. 2 (a).

2.2.4 Identification of the Orientation of Robots

Although each robot is specified by a bounding box shaped like a non-regular rectangle, the orientation of the robot can face either forward or backward. We capture the overhead image of the robot as a template and find a better matching orientation based on the template image. This make it so that we can recognize multiple robots simultaneously without any markers as shown in Fig. 2 (b). Even if we must use different kinds of robots, we only have to exchange the template.

2.3 Virtual Application

The virtual application carries out physical simulations, such as collision detection based on the results of the recognition program and then projects the simulation results on the field. Thus, the real robots can play a soccer game even with a virtual ball. In addition, the virtual application can communicate with real robots, and so the robots can receive the ideal positions of real objects without an accurate localization system.

2.3.1 Simulator

Our simulator is based on Haribote [7], the dynamic simulator for AIBO, developed by utilizing the Open Dynamic Engine (ODE) and the Open Graphical Library (OpenGL). Although Haribote has a virtual robot represented as the exact model of AIBO, in order to represent the robots recognized by the ceiling cameras, we utilize the minimum rectangular

(a) Extraction of contour (b) Identification of the orientation of robots

Fig. 2 (a) Results extracting robots as small rectangles and other objects as large rectangles. (b) Results of template matching. An arrow shows the orientation of each robot.

solid for covering the robot. The reason is that the rectangular model can more easily synchronize real robots so that the real robot can manipulate a virtual ball in our simulator. Of course, we can utilize the exact model in order to verify some physical motions in the simulator as well.

2.3.2 Sending of the Positions of Objects

The virtual application always sends the precise positions of the physical objects recognized by the recognition program to real robots through UDP. The robots can receive the positions of the ball and the other robots and replace noisy inputs with precise ones in their programs. This indicates that we can focus only on debugging of developed strategies in ideal situations in spite of using real robots. We may partially use those precise positions so that the augmented environment can get closer to real environments.

2.3.3 Projection of the Simulator on the Field

The virtual application projects the simulator on the field in order to visualize the positions of objects in the simulator. In addition, if we display some debugging information in the simulator, we can see the debugging information (e.g., the robot's position, the ball's position, and the robot's state) overlaid on the field instead of that in a console terminal. Although our projector projects data from an oblique perspective, this fact does not cause problems at all by utilizing the automatic trapezoidal correction in the projector. Figure 3 shows our simulator projected on the field by the projector. The upper robot is real, and the lower robot and the ball are virtual. In our system, both of the real robot and the virtual robot can manipulate the virtual ball.

2.4 Portable System

In this section, we show the portable version of our system

Fig. 3 Our simulator that is projected on the field.

that does not require ceiling projectors. The portable version allows us to take advantage of our system even in the situation that we cannot set up ceiling projectors, i.e., when we are at the competition site of RoboCup. The idea to achieve portability is by overlaying virtual objects with the soccer field not in the real world, but instead on the image captured by a web camera using ARToolKit [16]. We develop the position visualizer system as an instance of the portable version and then discuss its advantages and disadvantages.

2.4.1 ARToolKit

ARToolKit is a program library written in the C programming language that can be imported and used by Microsoft Visual Studio. It is used in creating Augmented Reality programs. Using a computer vision algorithm and through the utilization of OpenGL and a web camera it uses pattern recognition to display 3D objects onto designated markers stored within ARToolKit in real time. However, a user is not limited to using only these default patterns. ARToolKit

can calibrate itself to recognize custom patterns and save them. The only requirements are that the pattern be black with a white background, with a black border followed by a white border. Also to make it easier to detect, it is recommended to use patterns that are asymmetric and do not have fine detail on them.

ARToolKit can be programmed to use the markers to display 3D objects upon, or it can be programmed to use the markers to create a grid in which to display 3D objects within. In the case of using the markers as a grid, ARToolKit can treat the area between all markers as a coordinate system and then 3D objects can be placed at coordinates specified in the program.

2.4.2 Position Visualizer System

The position visualizer system uses ARToolKit's mutlimarker functionality to create a planar system using 6 markers and displays 3D objects onto the virtual field, which is overlaid with the real soccer field. Cones are used to represent the robots and a sphere is used to represent the ball. Since the robots send out data through User Datagram Protocol (UDP) the program listens to the port used in order to get field information, such as the robot positions and ball position. Using that data the 3D objects are placed onto the virtual field created by the 6 markers. When those 6 markers are placed around the soccer field, the virtual field will match the soccer field and the 3D objects will be placed where the robot believes it is positioned and where it thinks the ball is.

This position visualizer system has its advantages and disadvantages. One advantage is that it is portable since all it requires is a terminal, a web camera, and the printed markers. Also, having the virtual data overlaid in real time with reality makes it very easy to see how close the robot's data is to reality; whereas before a person would need to look at a screen with only the data and then visually compare that with the soccer field. However, it is sometimes difficult to see the whole field from the perspective of a single web camera. Therefore analyzing and debugging the positions and data of just a small section of the field would probably be more useful e.g., the goalie box as shown in Fig. 4. It was also determined that as long as there is one visible marker, the virtual field can still be created and the 3D objects displayed correctly. This is useful when other markers are obstructed from view for any number of reasons. On the other hand, this system is essentially a visual representation of data, so currently it does not have a way to easily recognize a robot's position on its own or a way to know the robot's current state or action. There is also more functionality that could be added to the program to make it more beneficial, such as having more data visually represented e.g., particles in the particle filter algorithm.

2.5 Contributions

The main merit of our system is that we can easily find which program exerts a bad influence to the whole program. For example, if we simulate the robot's position and ball's position, we can verify the validity of strategy programs by utilizing the precise output of localization programs. In the same way, if we simulate the beacons' positions, we can verify the validity of localization programs by utilizing the precise output of vision programs.

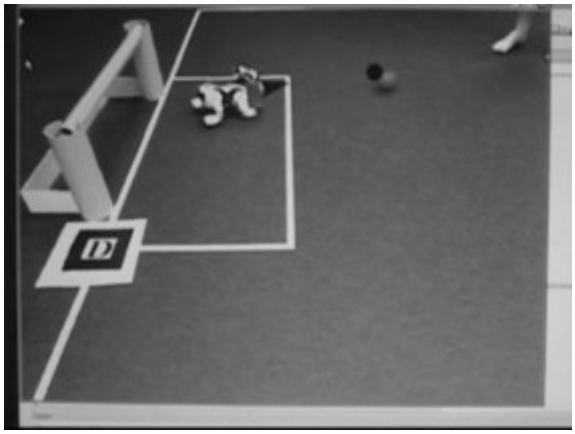


Fig. 4 Here the cone represents the estimated position the robot and the dark sphere represents the estimated position of the ball as the robot perceives it.

Another merit of our system is the visualization of debugging information by the projector. For example, we can visualize the particles in localization programs using a particle filter algorithm as well as the objects recognized by vision programs. We can also visualize the current state of strategy programs and even the error messages of programs as strings.

3. Autonomous Learning of Goalie Strategies

In this section, we address the learning of goalie strategies in a penalty kick in SPL as an application of our system. We first describe the rule of penalty kicks and the learning method of goalie strategies. Next, we experiment in our simulator to ensure the validity of the method, and finally experiment in our augmented environment using the learning results from the simulation.

3.1 Penalty Kick

In the rules for SPL, the penalty kick is carried out with one attacker and one goalie. If the ball goes into the goal within the time limit (1 minute), the penalty kick is deemed successful. If the time limit expires or if the ball leaves out the field, the penalty kick is deemed unsuccessful. In addition, if the attacker enters the penalty area then the penalty kick is deemed unsuccessful; if the goalie leaves the penalty area then a goal will be awarded to the attacker. The aim of our goalie is to acquire the best strategy to save the goal.

3.2 Learning Method

In this paper, we apply reinforcement learning algorithms [17]. Since reinforcement learning requires no background knowledge, all we need to do is to give the robots some appropriate rewards so that they can successfully learn goalie strategies.

The reinforcement learning process is described as a sequence of states, actions, and rewards,

$$s_0, a_0, r_1,\ s_1, a_1, r_2,\ \ldots, s_i, a_i, r_{i+1},\ \ldots,$$

which is a reflection of the interaction between a learner and an environment. Here, $s_t \in S$ is a state given from the environment to the learner at time t ($t \geq 0$), and $a_t \in \mathcal{A}(s_t)$ is an action taken by the learner for the state s_t, where $\mathcal{A}(s_t)$ is the set of actions available in the state s_t. One time step later, the learner receives a numerical reward $r_{t+1} \in \mathcal{R}$, in part as a consequence of its action, and finds itself in a new state s_{t+1}.

- **State** The state of our learner is represented by a list of 6 scalar variables $(r, \phi, V_v, V_h, X, Y)$. The variables r and ϕ represent the position of the ball on a polar coordinate and refer to the distance and angle from the robot to the ball estimated by our vision system. The variables V_v and V_h represent the velocity of the ball and refer to the vertical and horizontal velocity from the robot. The variables X and Y represent the relative position of the robot from its own side's goal and refer to the values of x-axis and y-axis on a right handed Cartesian coordinate system with its origin at the center of the goal, i.e., an absolute position (2700 mm , 0 mm), and a positive x-axis toward the center of the field, i.e., an absolute position (0 mm, 0 mm). In order to allocate the required minimum state space for our problem, we empirically limited the range of those state variables such that r, ϕ, V_v, V_h, X, and Y are in [0 mm, 1000 mm], [$-\pi/2$ rad, $\pi/2$ rad], [-50 mm, 0 mm], [-50 mm, 50 mm], [-500 mm, 1000 mm], and [-700 mm, 700 mm], respectively.
- **Action** The action of our learner takes one of the following 10 exclusive actions. One is *save* to stop an incoming ball, which performs the motion to spread out robot's front legs. The action *save* cannot be interrupted for 350 ms until the motion finishes. Another is *stay* to prepare for an opponent's shooting, which moves its head to watch the ball without walking. The others are 8 directional walking actions (i.e., vertically, horizontally, or diagonally) to intercept an opponent's shooting.
- **Reward** Our learner receives one of the following 5 kinds of rewards. A negative reward is also called a *punishment*.
  - *save_punishment* The punishment is -0.02, if the robot performs *save* action, since the time during *save* motion has the risk losing the ball.
  - *passive_punishment* The punishment is -0.0000001, if the robot performs an action other than *save* action, since the robot should take various actions in the initial phase of learning.
  - *lost_punishment* The punishment is -10, if the ball goes into the goal.
  - *save_reward* The reward is 0.5, if the robot stops the ball by *save* action, since it may be safest to save the goal.
  - *dist_reward* The reward is $1 - |ydist| / 112.5$, if the robot saves the goal during a penalty kick, since it may be the safest saving. The variable $ydist$ is the y-axis distance of the ball, which is limited in [-225 mm, 225 mm] based on the covering range 450 mm of *save* action. That is, *dist_reward* takes a value in [-1, 1] based on $ydist$.

Algorithm 1 shows the autonomous learning algorithm of goalie strategies used in our study. It is a combination of the episodic SMDP Sarsa(λ) with the linear tile-coding function approximation (also known as CMAC). This is one of the most popular reinforcement learning algorithms, as seen by its use in Kobayashi et al. [14]. In our experiments, we define the period from the starting to the ending of a penalty kick as one *episode*.

Here, F_a is a *feature set* specified by tile coding with each action a. In this paper, we use 6-dimensional tiling, where we set the number of tilings to 32 and the number of tiles to about 400000. We also set the tile widths of r, ϕ, V_v, V_h, X, and Y to a quarter of the ranges, i.e., 250 mm, $\pi/4$ rad, 12.5 mm, 25 mm, 375 mm, and 350 mm, respectively. The vector $\overrightarrow{\theta}$ is a *primary memory vector*, also known as a *learning weight vector*, and Q_a is a *Q-value*, which is represented by the sum of $\overrightarrow{\theta}$ for each value of F_a. The policy ϵ-*greedy* selects a random action with probability ϵ, and otherwise, it selects the action with the maximum Q-value. We set $\epsilon = 0.001$. Moreover, $\overrightarrow{e}$ is an *eligibility trace*, which stores the credit that past action choices should receive for current rewards. λ is a *trace-decay parameter* for the eligibility trace, and we set $\lambda = 0.9$. We set the *learning rate parameter* $\alpha = 0.18$ and the *discount rate parameter* $\gamma = 1.0$.

3.3 Experiments in Simulated Environments

We experiment in the simulated environment (or simulator) to ensure the validity of the algorithm in the previous section. In our simulator we do not treat the dynamics of robots, while the scales of the field, ball, and robots are set to the same values in real environments. Our virtual robot is represented as a rectangular solid, and if *save* action is performed then the width of the rectangular solid stretches out. State inputs have their ideal values without any noise, and action outputs are set to almost the same values as real robots. In experiments from now on, we assume that the attacker dribbles the ball to any point and then shoots it to the goal so as to confuse the goalie. Thus, we set the ball in the simulator to perform such behavior in the beginning of penalty kicks.

Algorithm 1: Algorithm of the autonomous learning of goalie strategies.

while *still not acquiring goalie strategies* **do**
  initialize penalty shootout settings.;
  $s \leftarrow$ a state observed in the environment;
  forall $a \in \mathcal{A}(s)$ **do**
    $F_a \leftarrow$ set of tiles for a, s;
    $Q_a \leftarrow \sum_{i \in F_a} \theta(i)$;
  end
  $lastAction \leftarrow$ an optimal action selected by ϵ-*greedy*;
  $\overrightarrow{e} \leftarrow 0$;
  forall $i \in F_{lastAction}$ **do** $e(i) \leftarrow 1$;
  $reward \leftarrow 0$;
  while *during a penalty kick* **do**
    do *lastAction*;
    if *lastAction* = *guard* **then**
      $reward \leftarrow save\_punishment$;
    else
      $reward \leftarrow passive\_punishment$;
    end
    $\delta \leftarrow reward - Q_{lastAction}$;
    $s \leftarrow$ a state observed in the environment;
    forall $a \in \mathcal{A}(s)$ **do**
      $F_a \leftarrow$ set of tiles for a, s;
      $Q_a \leftarrow \sum_{i \in F_a} \theta(i)$;
    end
    $lastAction \leftarrow$ an optimal action selected by ϵ-*greedy*;
    $\delta \leftarrow \delta + Q_{lastAction}$;
    $\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \alpha\delta\overrightarrow{e}$;
    $Q_{lastAction} \leftarrow \sum_{i \in F_{lastAction}} \theta(i)$;
    $\overrightarrow{e} \leftarrow \gamma\lambda\overrightarrow{e}$;
    forall $a \in \mathcal{A}(s)$ *s.t.* $a \neq lastAction$ **do**
      forall $i \in F_a$ **do** $e(i) \leftarrow 0$;
    end
    forall $i \in F_{lastAction}$ **do** $e(i) \leftarrow 1$;
  end
  if *lost the goal* **then**
    $reward \leftarrow lost\_punishment$;
  else
    $reward \leftarrow dist\_reward$;
    if *lastAction* = *guard* **then**
      $reward \leftarrow reward + save\_reward$;
    end
  end
  $\delta \leftarrow reward - Q_{lastAction}$;
  $\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \alpha\delta\overrightarrow{e}$;
end

We made our robot on the simulator learn goalie strategies in 2000 episodes, and the experiment took about 20 minutes. Figure 5 (a) shows the learning process in the simulated environment in terms of the success rate of saving. The success rate of saving means how many times the robot saved the goal in the past 200 episodes. The figure indicates that the success rate reached more than 90% at 2000 episode. Actually, we repeated the same experiment 5 times and got the average (and standard deviation σ) of the final success rates, 93.32% (σ=2.13). This result indicates that our virtual robot can successfully acquire a better goalie strategy in penalty kicks.

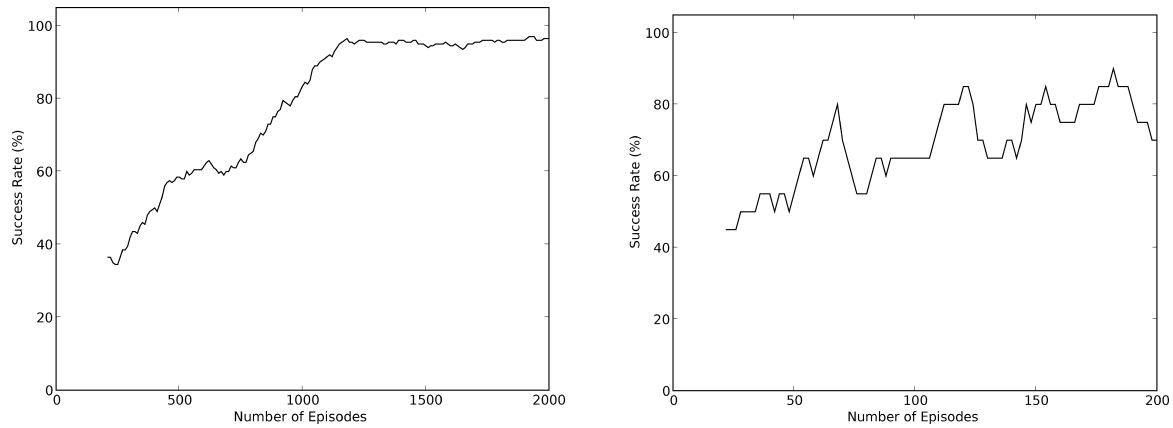
Figure 6 shows an acquired goalie strategy (i.e., a state-action mapping) after 2000 episodes. This intuitively visualizes all of the better actions that the learner will select at each state in the environment. The figure indicates that *save* actions shown by squares are mostly on the way to the destination of a kicked ball so as to intercept the incoming ball. Although walking actions shown by arrows have seemingly no regularity, looking at the arrows only around the squares, we find that the directions of the walking actions mostly face to the states with *save* actions, so that the robot can move to intercept the ball if possible. The states with the other walking actions (far from the squares) seem not to be learned yet. The reason is that our virtual robot never loses the ball in the simulator, and so it cannot go through the experience of those states. In the case of the penalty kick, this may not be a big problem, since the robot knows the fixed initial positions of the attacker and the ball in advance.

3.4 Experiments in Augmented Environments

In this section we experiment in our augmented environment, where we set the ball as simulated and project to the field. Since a virtual ball is easily manipulated by our system, our robot does not need to perform troublesome tasks such as restoring the ball, and this advantage easily enables the autonomous learning of goalie strategies. Moreover, our robot can receive the ideal positions of the ball and the robot itself from our system. This also means that we can experiment using real robots in real environments, even if some basic programs such as vision and localization systems are premature or unfinished.

We made our robot learn goalie strategies in 200 episodes, starting from the strategy acquired in the previous section, since it takes a lot of time and cost for real robots to perform a lot of actions. The experiment took about 45 minutes with 1 battery. Figure 5 (b) shows the learning process in the augmented environment in terms of the success rate of saving. In the initial phase of the process, the success rate was only about 50% despite the initial strategy whose success rate was more than 90% in the simulator. This suggests that there is a certain gap between simulated environments and real environments, especially in terms of quadrupedal locomotion. In the latter phase of the process, the success rate increased to about 80%, which may be the maximum value since real robots cannot walk as smoothly as virtual robots. This also suggests that our system can fill in such a gap. We repeated the same experiment 5 times and got the average success rates (and standard deviations σ) in the first 20 episodes and the last 20 episodes in 200 episodes, which were 50.0% (σ=10.0) and 77.0% (σ=7.48), respectively.

Although our system filled in a gap in terms of quadruped locomotion, there should be a gap in terms of ball and robot positions. This means that the strategies acquired in the augmented environment might not be work well in real environments. However, we take a stance that we should improve the estimation methods of those positions such as the Kalman filter. That is because the latter gap becomes smaller as the estimation methods become better, while the former gap tends to become bigger as quadruped locomotion becomes better, that is faster.



(a) In the simulated environment (b) In the augmented environment

Fig. 5 Experimental results in our simulated environment and augmented environment, whose success rates indicate how many times the robot saved the goal in the past 200 episodes and 20 episodes, respectively.

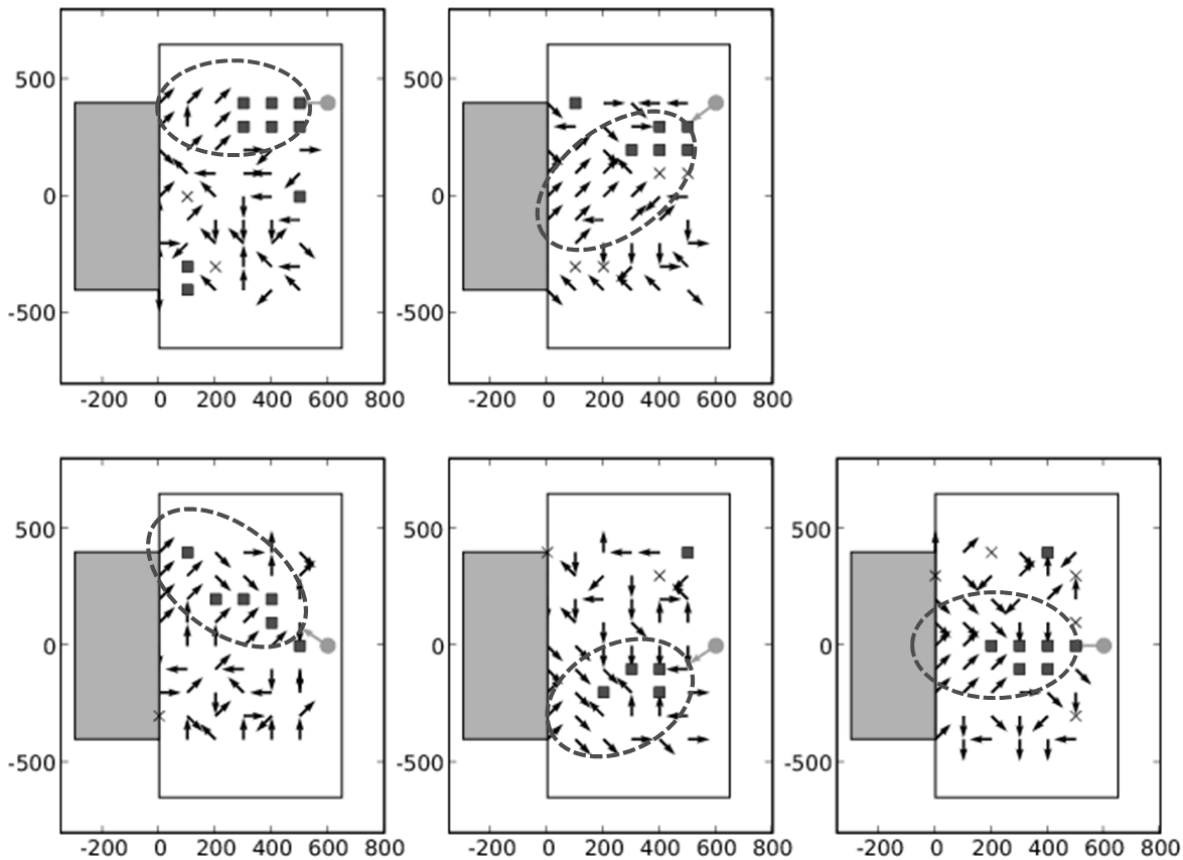


Fig. 6 State-action mapping that was acquired by the learning in the simulator. Each figure shows the left half field, and its coordinate shows the relative position of the robot from the goal, (i.e., *X* and *Y* in the state of our learner). Each pair of a circle and arrow shows the velocity of a ball, and each figure indicates which action our robot selects in the situation. A square, cross, and arrow show *save*, *stay*, and walking actions, respectively. In the area enclosed with a dashed circle in each figure, better actions are selected so as to intercept the incoming ball.

4. Conclusion

In this paper, we presented an interactive experimental environment to bridge the gap between simulated environments and real environments, and developed an augmented soccer field system to produce the environment by utilizing camera input and projector output. The significant point com-

pared with analogous work is that virtual objects are touchable in this system owing to projectors. We also developed the position visualizer as a portable version of our system. Moreover, we addressed the learning of goalie strategies in penalty kicks as an application of our system. By utilizing our system, our robot could autonomously learn and acquire better strategies without human intervention.

References

[1] K. Asanuma, K. Umeda, R. Ueda, and T. Arai, "Development of a simulator of environment and measurement for autonomous mobile robots considering camera characteristics," RoboCup 2003: Robot Soccer World Cup VII, LNAI, vol.3020, pp.446–457, Springer-Verlag, 2004.
[2] T. Ishimura, T. Kato, K. Oda, and T. Ohashi, "An open robot simulator environment," RoboCup 2003: Robot Soccer World Cup VII, LNAI, vol.3020, pp.621–627, Springer-Verlag, 2004.
[3] J.C. Zagal and J.R. del Solar, "UCHILSIM: A dynamically and visually realistic simulator for the RoboCup four legged league," RoboCup 2004: Robot Soccer World Cup VIII, LNAI, vol.3276, pp.34–45, Springer-Verlag, 2005.
[4] T. Laue, K. Spiess, and T. Röfer, "SimRobot — A general physical robot simulator and its application in RoboCup," RoboCup 2005: Robot Soccer Wold Cup IX, LNAI, vol.4020, pp.173–183, Springer-Verlag, 2006.
[5] M. Zaratti, M. Fratarcangeli, and L. Iocchi, "A 3D simulator of multiple legged robots based on USARSim," RoboCup 2006: Robot Soccer World Cup X, LNAI, vol.4434, pp.13–24, Springer-Verlag, 2007.
[6] Microsoft, "Microsoft robotics studio," 2007. URL: ⟨http://msdn.microsoft.com/robotics/⟩
[7] K. Takeshita, T. Okuzumi, S. Kase, Y. Hasegawa, H. Mitsumoto, R. Ueda, K. Umeda, H. Osumi, and T. Arai, "Technical report of team ARAIBO," Tech. Rep., ARAIBO, 2007. URL: ⟨http://araibo.is-a-geek.com/⟩
[8] M. Stilman, P. Michel, J. Chestnutt, K. Nishiwaki, S. Kagami, and J.J. Kuffner, "Augmented reality for robot development and experimentation," Tech. Rep. CMU-RI-TR-05-55, Robotics Institute, 2005.
[9] K. Sugawara, T. Kazama, and T. Watanabe, "Foraging behavior of interacting robots with virtual pheromone," Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004), pp.3074–3079, IEEE, 2004.
[10] M. Sugimoto, G. Kagotani, M. Kojima, H. Nii, A. Nakamura, and M. Inami, "Augmented coliseum: Display-based computing for augmented reality inspiration computing robot," SIGGRAPH '05: ACM SIGGRAPH 2005 Emerging Technologies, p.1, ACM, 2005.
[11] "Standard platform league official site," URL: ⟨http://www.tzi.de/spl/⟩
[12] "RoboCup official site," URL: ⟨http://www.robocup.org/⟩
[13] R. da Silva Guerra, J. Boedecker, N. Mayer, S. Yanagimachi, Y. Hirosawa, K. Yoshikawa, M. Namekawa, and M. Asada, "CITIZEN Eco-Be! league: Bringing new flexibility for research and education to RoboCup," Proc. Meeting of Special Interest Group on AI Challenges, vol.23, pp.13–18, 2006.
[14] H. Kobayashi, T. Osaki, E. Williams, A. Ishino, and A. Shinohara, "Autonomous learning of ball trapping in the four-legged robot league," RoboCup 2006: Robot Soccer World Cup X, LNAI, vol.4434, pp.86–97, Springer-Verlag, 2007.
[15] M. Piccardi, "Background subtraction techniques: A review," Systems, Man and Cybernetics, 2004 IEEE International Conference on, vol.4, pp.3099–3104, 2004.
[16] "ARToolKit official site," URL: ⟨http://www.hitl.washington.edu/artoolkit/⟩
[17] R.S. Sutton and A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998.

Appendix: Movies

All movies of the demonstration of our system and the experiments of the learning in our system are available online †.

†URL: ⟨http://www.shino.ecei.tohoku.ac.jp/~kobayashi/movies.html#goalie⟩

Hayato Kobayashi received his master degree in Department of Informatics from Kyushu University in 2007. He is a doctoral course student at Graduate School of Information Sciences, Tohoku University. He is also a JSPS research fellow. His research interests include machine learning/discovery and their applications to real robots. He is a student member of IPSJ, JSAI, and RSJ.

Tsugutoyo Osaki received his master degree in Graduate School of Information Sciences from Tohoku University in 2008. He joins Toshiba Corporation in 2008. His research interests include machine learning and their applications to real robots.

Tetsuro Okuyama received his master degree in Graduate School of Information Sciences from Tohoku University in 2008. He joins Panasonic Corporation in 2008. His research interests include computer vision and image processing.

Joshua Gramm is currently a Computer Science undergraduate at the University of California, San Diego. His field interests include video game design and hardware.

Akira Ishino received his master and doctor degrees in Graduate School of Engineering from Hokkaido University in 1995 and 1999, respectively. From 2006 to 2008, he was an Assistant Professor of Graduate School of Information Sciences, Tohoku University. He is currently a software engineer, Google Corporation, Japan. His research interests include machine learning, string processing and search algorithms. He is a member of IPSJ and JSAI.

Ayumi Shinohara received his master and doctor degrees in Information Systems from Kyushu University in 1990 and 1994, respectively. Presently, he is a Professor of Graduate School of Information Sciences, Tohoku University. His research interests include machine learning/discovery and string processing algorithms. He is a member of IPSJ and JSAI.