Virtual RoboCup: Real-Time 3D Visualization of 2D Soccer Games

Bernhard Jung, Markus Oesker, and Heiko Hecht

Universität Bielefeld, Germany
http://www.TechFak.Uni-Bielefeld.DE/techfak/ags/wbski/3Drobocup/

Abstract. Virtual RoboCup is a real-time 3D visualization tool for 2D simulated soccer games as played in the RoboCup simulation league. Players are modeled as anthropmorphic figures that are animated step-keepingly with the underlying 2D simulation. Important aspects of player animation concern the generation of natural 3D player movements and realistic player-ball interactions during kicks. A key contribution of Virtual RoboCup is its novel approach to task-level animation in which task-level directives for 3D animation of anthropomorphic characters are generated via on-line classification of fast paced 2D simulation data. As further contribution, we investigated to what extend observers perceptually process the level of detail in naturalistic character animations. A psychological experiment was designed to test the effectiveness of 3D body animation. Although observers failed to notice differences in animation detail, clear effects of character animation on perceived skill were found. We conclude from these results that is is very well justified to spend valuable computational resources on richness of detail in realtime character animation.

1 Introduction

Recent advancements in computing power and graphics rendering technology have facilitated the development of many applications (e.g. in computer games, ergonomic evaluation of virtual prototypes, multi-user virtual worlds, etc.) involving interactive animation of 3D human-like figures [7, 1]. Such animated figures are typically controled via task-level directives which are translated by the animation system into appropriate geometric transformations at the graphical level [12, 8]. Typical task-level commands for interactive animations are, for example, *walk from A to B* or *kick the ball in direction* α. Other work has explored cases where task-level animation commands originate in instructions by a human, e.g. using natural language [2, 11], or in the planning processes of autonomous agents situated in the virtual environment [10, 3, 9]. In this article, we explore a third source of input to task level control of animated 3D figures: 2D state information about a simulated soccer game.

Virtual RoboCup is a real-time 3D visualization system for simulated 2D soccer games as played in the RoboCup simulation league [5]. The 2D soccer

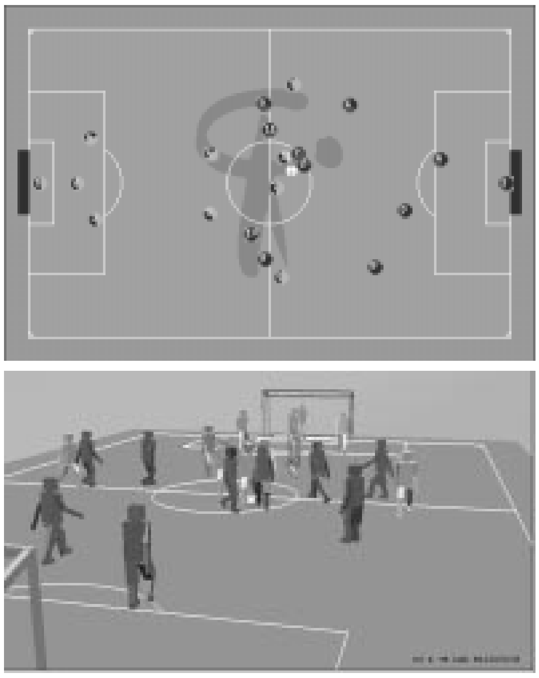

Fig. 1. RoboCup provides a 2D environment for simulated soccer games (top). Given 2D input from the soccer server, Virtual RoboCup generates real-time 3D visualizations where players are animated as anthropomorphic figures (bottom).

simulator and a 2D visualization program are provided by the RoboCup organization (Figure 1 top). Virtual RoboCup adds anthropomorphic figures to the 2D simulation and animates the soccer game in real-time (Figure 1 bottom). Virtual RoboCup classifies 2D information about the soccer game into task-level action commands which are used to animate the 3D visualized soccer players. Special complexities of this approach arise (a) from the high frequency in which task level commands need to be generated, (b) from the relatively high number - there are 22 players to a soccer game - of animated figures, and (c) from the frequent interactions between the animated players among each other and the ball.

In developing Virtual RoboCup, high emphasis was placed on generating natural and physically plausible animations of soccer games. More concretely, we focussed on the following goals:

- Real-time animation: The 3D animation is generated on the fly; the computation of 3D simulation data keeps step with 2D input from the RoboCup server.
- Natural, human-like movements of players: 3D players, as opposed to their circle representations in the 2D RoboCup server, have legs and they must

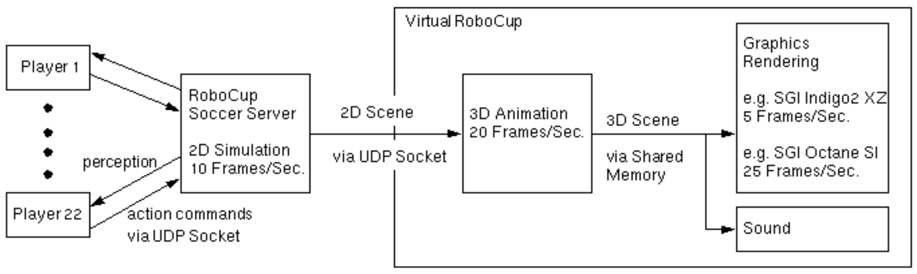
Virtual RoboCup
Player 1
Player 22
perception
action commands via UDP Socket
RoboCup Soccer Server
2D Simulation 10 Frames/Sec.
2D Scene
via UDP Socket
3D Animation 20 Frames/Sec.
3D Scene
via Shared Memory
Graphics Rendering
e.g. SGI Indigo2 XZ 5 Frames/Sec.
e.g. SGI Octane SI 25 Frames/Sec.
Sound

Fig. 2. The RoboCup server defines a 2D environment for simulated soccer games; player actions are controled by independent processes. Virtual RoboCup is realized through several, asynchronous processes in order to allow for a step-keeping 3D visualization of 2D RoboCup games on different hardware platforms. The 3D animation process adds intermediate states to the soccer simulation, generating 3D scenes at a fixed rate of 20 frames per second. The graphics rendering process always presents the most recent 3D scene; its update frequency depends on the underlying graphics hardware. A sound process generates acoustic feedback about successful kicking actions.

use them when moving on the field. Animation of players' movement should be fluent, without discontinuities. Furthermore, the 3D players' stepping frequency should be no faster than human stepping.

- Physically realistic kicking actions: In order for a player to kick the ball, contact between the player's foot and the ball must be established. A kick, in contrast to its instantaneous nature in the RoboCup server, is a temporally extended action that lasts over several animation phases.

Thus, in addition to the real-time requirement, Virtual RoboCup is mainly concerned with the addition of articulated body models and the dynamics of players' footwork when running and kicking. While the 2D information from the RoboCup server provides some constraints on these tasks, it sometimes also admits *unnatural* movement of players (see next section). Virtual RoboCup makes conservative attempts to "smoothen" unnatural player behavior into movements more consistent with human biomechanics. In general, however, the 3D animations produced by Virtual RoboCup accurately reflect the 2D game states of the RoboCop simulation.

2 System Architecture

The system architecture of Virtual RoboCup reflects the aforementioned criteria of real-time capability and naturalness of players' running and kicking actions. Figure 2 summarizes the 3D visualization architecture of Virtual RoboCup and its relationship to the 2D soccer simulation in the RoboCup server. RoboCup soccer games are distributed simulations consisting of up to 22 players, realized as

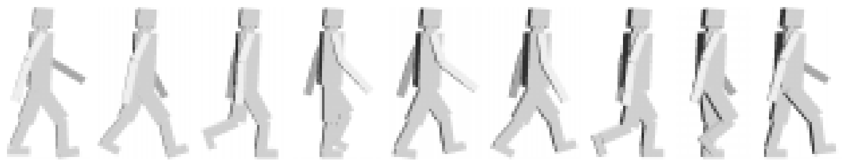

Fig. 3. Some locomotion keyframes for Virtual RoboCup players.

independent control processes, and a central server that maintains the current simulation state. The simulation environment defined by the RoboCup server is two-dimensional, i.e. players and ball are represented as circles. Every 100 ms, the RoboCup server generates a snapshot of the game state describing the current positions of players and ball as well as some additional information about players' kicking actions and scored goals. These snapshots constitute the input of visualization systems such as Virtual RoboCup.

The system architecture of Virtual RoboCup consists of three asynchronous processes (communicating via shared memory) for 3D animation, graphics rendering, and sound generation. The main rationale behind the asynchronous computation of 3D animation data and graphics rendering is to avoid having the (usually) faster simulation process wait for the (usually) slower rendering process. For example on an SGI Indigo 2 XZ platform, a rendering rate of 5 frames per second is achieved for Virtual RoboCup. If 3D animation and rendering were synchronized, the 3D animation step rate would also slow down to 5 frames per second, thus falling behind the input data arriving at the rate of 10 frames per second. With asynchronous animation and rendering, the rendering process visualizes the most recent 3D animation state, possibly dropping some intermediate states. If a platform with faster graphics capabilities is used, e.g. an SGI Octane SI, all frames can be rendered. The asynchronous computation thus ensures that the 3D visualization – independent from the specific graphics hardware used – keeps step with the 2D simulation.

When generating the 3D animation of the soccer game, Virtual RoboCup adds intermediate states to the original 2D RoboCup simulation, such that the internal simulation of Virtual RoboCup runs with twice the speed of the RoboCup simulation. One reason for this speed-up is that a more fluent visualization can be generated. Another, deeper reason has to do with the way that kicking actions are calculated in the RoboCup simulation: When the RoboCup soccer server establishes that, in a simulation cycle t_i, a kicking attempt of a player was successful it will also, in the same cycle, calculate a new ball position reflecting the effect of the kicking action. Thus, the soccer server might generate a new game state description for simulation cycle t_i where the ball is outside of the player's kicking range yet annotate this scene symbolically that the player has kicked the ball. Therefore, if in the kicking action contact between the player's foot and the ball is to be visualized, then the time of contact must lie before t_i, yet after the preceding simulation cycle t_{i-1}, hence in some intermediate state.

Finally, the 3D animation is slightly (including time for graphics rendering less than 0.5 seconds) delayed as compared to the original simulation. This time delay is used, for example, for continuous animation of kicks (including preparing frames for change of supporting leg, leg swinging), that are treated as instantaneous events in the RoboCup server. Also, the 2D simulation in the RoboCup server is an abstract approximation of real soccer games that sometimes allows for unnatural, or even biomechanically impossible player movements. For example, with repeated "dash-turn" commands, 2D players can zig-zag across the field with up to 5 direction changes per second. Similarly, with repeated "turn" commands, 2D players can perform up to 5 pirouettes per second. To make the 3D players' movements appear more human-like, Virtual RoboCup examines the temporal context of player movements and smoothens sharp direction changes by averaging over several simulation cycles. As a last example, with quickly repeated "kick" commands, players can 'hyper-kick' the ball in consecutive 100 ms simulation cycles. By inspecting the temporal context of kicks, such repeated, instantaneous kicks in the 2D simulation are merged into one, yet temporally extended kicking action in Virtual RoboCup.

3 Body Model

The body model of Virtual RoboCup players is an articulated, anthropomorphic structure that consists of 15 segments representing the torso, legs, arms, neck and head. Body segments are shaped as boxes so as to allow for efficient graphics rendering (as a welcome side effect the players also appear more "robotic"). As body size, a height of 5 meters was chosen so as to enable kicking actions within 2 meter range of 2D RoboCup players (2D RoboCup players have a diameter of 1.6 meters!).

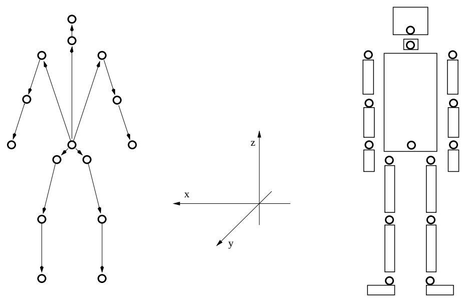

Fig. 4. The hierarchical body model of Virtual RoboCup players.

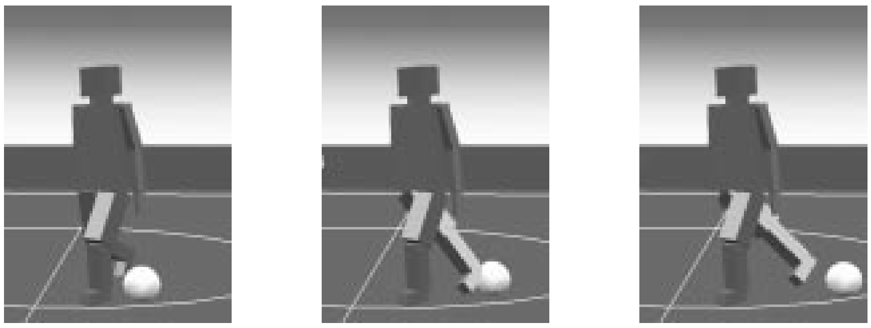

Fig. 5. Animation of kicking actions: Preparation – contact – follow-through.

4 Movement Animation

Virtual RoboCup tries to optimize animation of the 3D soccer players' running in two dimensions: First, players' movements should appear as natural as possible. For example, each visualized step of 3D players will extend over several cycles of the underlying 2D simulation. And second, generation of the 3D animation must occur in real time, keeping up with the 100 ms update frequency of the 2D simulation. While physical based simulations of human running result in highly realistic animations, e.g. [4], they are not yet computable in real-time. To meet the real-time requirement, Virtual RoboCup uses a keyframe based approach for animating the players' forward movements.

As foundation for the real-time movement animation, 30 locomotion keyframes are defined. The cyclic keyframe sequence shows a 3D player performing a step with the right leg followed by s step with the left leg (see Figure 3). The whole sequence will move the player 4.10 meters forward. The keyframe table also annotates each keyframe with the player's position gain as compared to the first frame of the sequence.

As the soccer players move on the field with changing speed but the animation proceeds with a fixed frequency, animation of the players' movements cannot simply consist of replaying the predefined keyframe sequence in standard order. Instead, the generation of the players' body postures during running animations also accounts for the player's position gain in an animation cycle: for a fast moving player, some intermediate keyframes might be dropped, whereas for a very slowly moving player, even the same keyframe might be used in consecutive animation cycles. Also, the 30 predefined keyframes are in certain cases, especially for very slowly moving players, insufficient to capture subtle changes in a player's running posture. Therefore, the players' animation postures are not limited to the predefined keyframes but calculated by interpolating between the predefined keyframes (*"inbetweening"*).

The algorithm for computing a player's body posture during running takes as input the 2D state information from the RoboCup server, or more concretely, the player's 2D position and orientation in the 100ms simulation cycle that is to be visualized as well as 2D state information from two preceding and two following simulation cycles. The animation process also maintains some additional internal state information about the players including their current posture (conceptually, a posture is represented as float within the range [0, 4.10] that 'indexes' into the keyframe table). The output of the algorithm is the 3D player's position, orientation and body posture for the next 50 ms animation cycle. The movement animation is computed in the following way:

1. The target position and orientation of the player for the next animation cycle are calculated from the 2D input data. If the animation cycle corresponds to a simulation cycle, the target position equals the player's position in the RoboCup simulation. If the animation cycle lies inbetween simulation cycles, the target position is calculated through linear interpolation between the player's positions in neighbouring cycles. The player's orientation is calculated by averaging over several simulation cycles, thus smoothening sharp direction changes.
2. To generate the player's 3D pose, first an 'posture index' into the keyframe table is computed by adding the position gain as compared to the last animation cycle to the player's posture index in the previous animation cycle. Then two frames f_- and f_+ are selected from the keyframe table that are closest to the new posture index. These frames are weighted according to their respective proximity to the new posture index and the player's actual body posture is generated by weighted interpolation between the two frames f_- and f_+ (inbetweening).

The running animations produced by this method appear especially natural in cases where 2D RoboCup players perform steady forward movements without sharp direction changes. One limitation of the current implementation is that the 3D players cannot yet come to a stillstand with both feet on the ground. In cases where the 2D simulation allows player movements that are per se impossible to perform given the constraints of human biomechanics (e.g. the zig-zagging and pirouetting behaviours described in Section 2), we had to make a choice, whether to stick with the original 2D running paths or to replace them with more natural paths. As the purpose of Virtual RoboCup is the visualization of 2D RoboCup games (and because improving on the 2D input data is in general a nontrivial, time consuming task) we decided to correct unnatural player movements only very cautiously. In particular, sharp direction changes of the players are smoothened by averaging a player's body orientation over several simulation cycles. The player's head is however always oriented according to the player's actual direction in the RoboCup server. Further improvements of the movement animation might involve the definition of several keyframe sequences for player movements in different speeds.

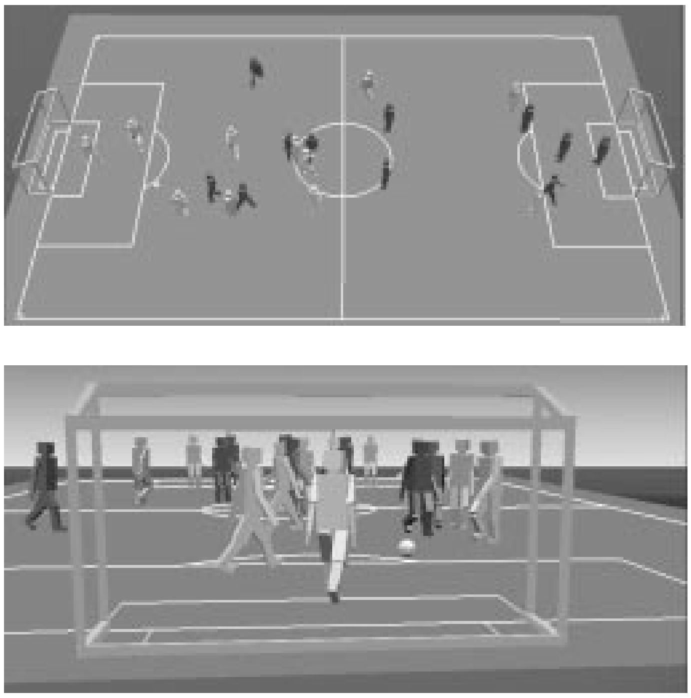

Fig. 6. A scene from the RoboCup'97 simulation league final, seen from two different perspectives.

5 Kicking Animation

For human soccer players (and human-like 3D players), a kick is fairly complex skill, involving e.g. selection of a foot to kick the ball with, approach of the ball such that the non-kicking foot gives enough support to keep the player balanced, leg swinging and orienting of the foot such that the ball is kicked in the intended direction, and so on. In contrast to that, the kick-model in the 2D RoboCup simulation is rather abstract: if close enough to the ball, a player can kick the ball in any direction. Furthermore, the RoboCup simulation treats kicks as instantaneous events and it is possible for single players to kick the ball in several consecutive 100 ms simulation cycles. Virtual RoboCup aims at visualizing the players' kicking actions as natural as possible. While not all details of human movements during ball kicking are reenacted, kicking actions are visualized as extended sequences: In the preparation phase, the player's leg swings towards the ball. In the culmination phase, contact between the player's foot and the ball is established. In the final follow-through phase, the leg keeps swinging in the direction of the kick (see Figure 5).

The RoboCup simulation allows a player to kick the ball anywhere within a distance of 2 meters and in any direction. A keyframe-based approach for animation of kicking actions is thus not feasible as it would require the definition of a too large amount of keyframe sequences to cover all free parameters. Instead, animation of kicking actions uses a inverse kinematics approach to guide the foot towards the ball.

The kicking animation is triggered whenever the RoboCup simulation reports a successful kicking attempt of a player. Further constraints on the kicking task, such as time, position, and direction of the kick are extracted from 2D game state information of several consecutive simulation cycles. The following steps are used to generate the kicking animation:

1. The direction in which the ball is kicked is calculated from the 2D input data. This is also the direction in which the kicking foot will approach the ball.
2. Based on the kicking direction and position of the ball w.r.t. the player, the foot to kick the ball with is selected. Foot selection also ensures that the player's legs don't overlap during the kicking sequence.
3. If necessary, the player is slightly repositioned such that a kick in the correct direction is possible. This might also involve a change of the supporting leg.
4. Using inverse kinematic techniques, the player's posture at ball-foot contact as well as at preparing and follow-through postures are calculated. These postures differ in the angles of hip, knee, and ankle joints. Computation of inverse kinematics uses a simple geometric approach.

An animated kick usually lasts for four animation cycles: Two preparation cycles, during which the foot is moved towards the ball, one contact cycle, and one follow-through cycle. For the reasons detailed in section 2, the contact cycle always falls in an animation phase between the RoboCup simulation cycles. If the RoboCup simulation reports kicks of the same player in consecutive cycles ('hyper-kick', see section 2), only the first kick is animated but with an prolonged follow-through phase; thus, visualization of 'hyper-kicks' is another example, where the 3D animation slightly deviates from 2D input data in order to make the players' movements appear more natural.

In Virtual RoboCup, players can kick the ball in any direction (see Figure 5 for a sideways kick); furthermore, players can kick the ball equally well with both feet. Although player animation based on inverse kinematics is computationally more expensive than keyframe based methods, the 3D visualization still meets the real-time requirement. This is due to the geometric (i.e. closed-form, non-iterative) approach for inverse kinematics calculation but also to the fact that usually at most one player performs a kick per simulation cycle. In parallel to the visual presentation of a kick, Virtual RoboCup also generates a characteristic sound to give the observer a fuller impression of the kicking action. For the time of a kicking animation, animation of the players' running movements is suppressed. Future work might involve improving the animation of transition phases between running and kicking.

Team	Goals
Sopra1	34:00
Sopra3	14:10
Sopra4	03:10
Krislet	00:31

Table 1. Goals scored by soccer teams during a competition.

6 A psychological experiment on human perception of animation detail

Detailled animation of 3D articulated body models is in principle desirable but it is also a highly resource-intensive task. It becomes particularly critical in 3D visualizations of multiple characters in real-time game sequences, such as Virtual RoboCup. Only if observers perceptually process (not necessarily consciously) visually presented animation details can it be justified to spend valuable resources on their computation. To test the influence of animation style on the observers' judgments of the capabilities of RoboCup simulation league soccer teams, we designed an experiment that allowed us to contrast the level of perceived playing skill with richness of detail in character animation.

First, a factor of objective skill level was created. Four teams were selected to span a large range of accomplishment. The teams' playing skills ranged from a tournament winner in 1998 to a team that was a few years back in evolution. All possible matches between the four teams were taken, also considering what team was on the left and the right side of the soccer field. From the resulting 12 recorded simulation league soccer matches attack sequences were isolated. They were selected such that always one goal was scored toward the end of the sequence. In all cases were the players and the ball represented such that a meaningful judgment of skill could be made. The 12 sequences were fully crossed with four different levels of animation detail, resulting in a total of 48 sequences. For each sequence observers were asked to specify the level of skill of both participating teams separately on a linear scale between zero and twelve. We were thus able to contrast the level of playing skill with the degree of character animation.

6.1 Design, Stimuli, Apparatus, and Procedure of the Experiemnt

First, a factor of objective skill level was created. Four teams were selected to span a large range of accomplishment. The teams' playing skills ranged from a tournament winner in 1998 to a team that was a few years back in evolution. Four clients (player agents) with known and heterogeneous abilities were used. A team consisted of five instances of one of the clients. Thus, each team consisted of five identical players controlled by the same algorithm but starting at different positions on the field. As main criteria for selection of clients we required that they were objectively discriminable by means of scored goals. Table 1 gives an insight into the selected teams' performance as exhibited during a competition. Three teams had participated in a competition that took place 1998 at the University of Bielefeld, Germany, and had reached the first, third and fourth place. The simple client named Krislet contributed by Kryzsztof Langner was taken from the Internet [6]. All four clients are implemented in Java, and work well with Soccerserver version 3.28, that was used for the experiment. Three competitions were recorded in which each of the four teams played against each of the others.

From the recorded soccer games we cut sequences of 20 seconds duration, which corresponds to 400 animation cycles. Each sequence showed a promising attack, which was defined as driving the ball in the direction of the opponent team's goal or at least the attempt thereof. Whenever possible we chose sequences that contained the scoring of a goal. The selected set of sequences contained scenes of all twelve possible combinations of teams. For each team an attack during a match against each of the other three teams was included.

We created four different animation levels by suppressing some features of the animation. Conditions were as follows: 1. No animation of running or kicking, 2. animation of running only, 3. animation of kick actions only, 4. running and kicking were both animated.

Each of the above sequences was presented four times using the different levels of animation detail. The resulting 48 sequences were presented in random order. The defending team was always named A, the other one B. The observer's point of view corresponded to a position near the corner to the right of the defender's goal. The direction of gaze was directed at the ball. This presentation ensured that observers could not identify teams except by the clients' actions.

The experiment was carried out as a fullscreen application on an SGI Indigo II XZ machine with a 21" monitor. For the experimental session the digitally recorded sequences were presented at a frame rate of 7 Hz[1]. The refresh rate of the monitor was 72 Hz. Eight student observers (4 men, 4 women) were paid for their participation. After viewing the 20 second sequence each observer had the opportunity to review the entire sequence if so desired. Then she was asked to first decide which of the two teams was more apt and skillful in its overall play. Once this decision had been made they had to assign a grade to each team. A grade of 0 corresponded to pitifully poor skill, a grade of 12 to exceedingly adept. About 10 practice trials were randomly selected from the pool of trials and presented to familiarize observers with the task.

6.2 Results and Discussion

After all data had been collected participants were asked what they thought distinguished the teams. They were also asked whether they had noticed any

[1] Rendering frequency would have been only 5 frames/second for a standard game with 22 players.

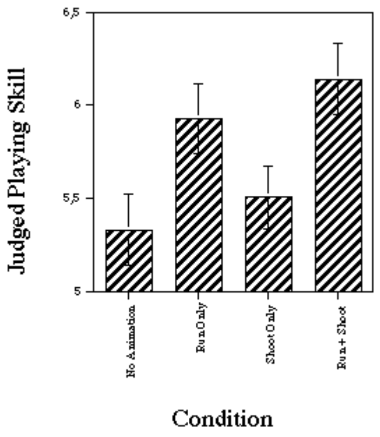

Fig. 7. Skill ratings averaged over all perfomance variations, plotted by degree of character animation. Error bars indicate standard errors of the mean.

changes in the animation of the bodies between trials. Amazingly, none of the observers reported changes in the body model. Even when directly asked whether in some trials players had moved or shot differently observers failed to report differences in animation. Observers fairly reliably recognized the objective skill of the teams. In 75.1 % of all cases they gave the higher ranking team (see Table 1) also a higher skill grade. The skill grades assigned to the different teams were correlated positively with their objective skill ($r = .51$, $p < .0001$). Thus, as expected, the main strategic differences produced by the clients were reflected in the judgments.

For the purposes of analyzing the effects of the unnoticed changes in character animation, the grade scores were entered into a repeated measures analysis of variance (ANOVA) with four levels of animation as independent factor. For the dependent variable, the average judged grade for both teams of a given sequence were computed. Note that the animation manipulation was always the same for both teams. For a perfect observer there should obviously be no differences for games that are identical except for the animation of all players. Thus, the ANOVA only reflects the influence of character animation. A significant main effect was found for this factor ($F(3, 21) = 3.28, p = .041$). As visible in Figure 7, the full animation was judged to be significantly more "skillful" than the absence of all character animation and also received better ratings than the animation of the shooting action only. The difference between no animation and shooting action only was not significant. Neither did the slightly better rating of full animation compared to running only reach significance.

Everything else controlled for, the fact that human observers failed to notice the animation manipulation did not prevent the animation to influence their judgments. A team whose characters are animated in their running and shooting action are judged to be more skillful. The running action tended to be most important in this context. These findings reveal that it is very well justified to spend valuable computational resources on richness of detail in real-time 3D character animation. They also reveal that explicit judgments, such as obtained by questionnaires or by mere inspection of the displays, are insufficient to asses the importance of level of detail.

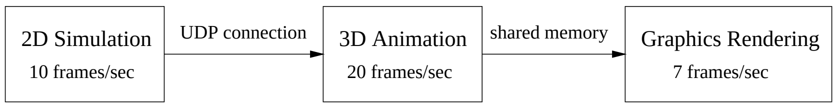

Fig. 8. The animation process generates 3D scenes at a fixed rate of 20 frames per second. Rendering speed depends on the underlying graphics hardware.

The fact that adding shooting detail to the running animation had very little effect on the results indicates that the limit of meaningful level of detail may have been reached. The poor effect of kick animations may not originate in the limits of observers' perceptual capacities, but may instead be caused by features of the animation itself. Soccer players are running nearly continously for the time of a match while shooting actions are performed only at times and by single players. Moreover, our animation technique may be suboptimal for visualisation of short and accentuated events like kicking actions. Animation is synchronized with incoming simulation data but unsynchronized with the rendering process (see Figure 8). As a consequence the 3D animation keeps step with the 2D soccer simulation. However, some simulation steps might not be visualized if rendering is slow. This loss may result in a significant effect on observer's perception since animation of kicking actions lasts only between four and six simulation cycles and contact to the ball is established only for the time of a single step. However, as disadvantagous as multiprocessing seems to be in this context it is indispensable. A linkage of the processes would result in a worse and flickering visualization.

7 Conclusions

We have described Virtual RoboCup, a real-time 3D visualization tool for simulated 2D soccer games. The soccer players are visualized as anthropomorphic characters whose running and kicking actions are animated in a natural fashion. Virtual RoboCup represents a novel kind of task-level animation system in which task-level commands are generated by classification of fast paced 2D simulation data. By visualizing soccer games with 22 players, Virtual RoboCup also demonstrates that real-time animation of a high number of human-like like characters is feasible with today's computing technology.

Virtual RoboCup is intended for 'live'-visualization of on-going 2D RoboCup simulation league games. Many of the design choices reflect this need for real-time capability, such as the simple body models of the 3D players, extensive use of keyframing and the asynchronous computation of 3D animation data and graphics rendering. Measurements of processing time show that the computation of the 3D player animation can easily keep up with input from the RoboCup Server. Rendering times, on the other hand, highly depend on the underlying graphics hardware (see section 2).

Besides the real-time requirement, important design goals of Virtual RoboCup included a high degree of naturalness in the players running and kicking animations. As the computation of realistic animations is a resource intensive task, we performed an experiment to test whether the effort for detailed figure animation is worth its while. Results show clear effects of character animation on perceived skill although observers were unaware of alterations in animation detail. Detail is processed unconsciously.

By virtue of being a 3D visualization, Virtual RoboCup offers, as compared to 2D visualizations, a more life-like presentation of RoboCup games. Soccer games can be watched from any angles, including the perspectives of the virtual players (figure 6). Moreover, Virtual RoboCup visualizes certain aspects of RoboCup games, e.g. the players' kicking attempts, that cannot be experienced in 2D visualizations. As alternative to standard desk-top displays, we have also ported Virtual RoboCup to the Responsive Workbench, hardware-software unit that projects stereoscopic 3D graphics on a translucent tabletop. Another aspect of the soccer game not presented in the the 2D visualization, namely visualization of the players' changing stamina states is a desirable candidate for further development.

References

1. N.I. Badler, C. Phillips, and B. Webber. *Simulating Humans: Computer Graphics Animation and Control.* Oxford University Press, New York, NY, 1993.
2. N.I. Badler, B.L. Webber, J. Kalita, and J. Esakov. Animation from instructions. In N.I. Badler, B.A. Barsky, and D. Zeltzer, editors, *Making them Move: Mechanics, Control, and Animation of Articulated Figures*, pages 51–93. Morgan Kaufmann, San Mateo, CA, 1991.
3. B. Blumberg and T. Galyean. Multi-level direction of autonomous creatures for real-time virtual environments. In *Computer Graphics Proceedings, SIGGRAPH-95*, 1995.
4. J.K. Hodgins. Three-dimensional human running. In *Proc. of the IEEE Conference on Robotics and Automation*, 1996.
5. H. Kitano, Tambe, P. M., Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, and M. Asada. The RoboCup synthetic agent challenge '97. In *Proc. of the 15th International Joint Conference on Artificial Intelligence, IJACI'97*, 1997.
6. Kryzsztof Langner. Krislet, a sample client for robocup simulation league, 1997. http://ci.etl.go.jp/~noda/soccer/client/index.html, 23 March 1999.
7. N. Magnenat-Thalmann and D. Thalmann. Complex models for visualizing synthetic actors. *IEEE Computer Graphics and Applications*, 11:53–59, 1991.
8. N. Magnenat-Thalmann and D. Thalmann. Computer animation. In *Handbook of Computer Science.* CRC Press, 1996.

9. K. Perlin and A. Goldberg. IMPROV: A system for scripting interactive actors in virtual worlds. In *Proc. SIGGRAPH'96*, pages 205–216, 1996.
10. D. Terzopoulos, X. Tu, and R. Grzeszczuk. Artificial fishes with autonomous locomotion, perception, behavior, and learning in a simulated physical world. In *Artificial Life IV: Proc. Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 17–27, Cambridge, MA, 1994.
11. I. Wachsmuth, B. Lenzmann, T. Jörding, B. Jung, M. Latoschik, and M. Fröhlich. A virtual interface agent and its agency. In W.L. Johnson, editor, *Proceedings of the First International Conference on Autonomous Agents*, pages 516–517. ACM Press, 1997.
12. D. Zeltzer. Motor control techniques for figure animation. *IEEE Computer Graphics and Applications*, 2(11):53–59, 1982.