An Entertainment Robot for Playing Interactive Ball Games

Tim Laue[1], Oliver Birbach[1], Tobias Hammer[2], and Udo Frese[1]

[1] Deutsches Forschungszentrum für Künstliche Intelligenz,
Cyber-Physical Systems, Bremen, Germany
{Tim.Laue,Oliver.Birbach,Udo.Frese}@dfki.de

[2] Institute of Robotics and Mechatronics, DLR, Germany
Tobias.Hammer@dlr.de

Abstract. This paper presents a minimalistic robot for playing interactive ball games with human players. It is designed with a realistic entertainment application in mind, being safe, flexible, reasonably cheap, and reactive. This is achieved by a clever, minimalistic robot design with a 2 DOF roll tilt unit that moves a bat with a spherical head. The robot perceives its environment through a stereo camera system using a circle detector and a multiple hypothesis tracker. The vision system does not require a specific ball color or background structure. The paper motivates the proposed robot design with respect to the above mentioned requirements, describes our solution to the tracking, calibration, and control issues involved and presents indoor and outdoor experiments where the robot bats balls tossed by different players.

1 Introduction

RoboCup Soccer has been founded as a basic research endeavour, as "an attempt to foster AI and intelligent robotics research by providing a standard problem where a wide range of technologies can be integrated and examined" [15]. However, unlike other basic research questions robot soccer is easily understood by the general public making the RoboCup competitions both a scientific and a public event. As Kitano said, a "publicly appealing but formidable challenge". This unique combination also motivates other "sport robotics" research activities, such as ball catching [5]. Now, being a basic research program, RoboCup soccer and other sport robotics activities are far from actual applications, they only contribute indirectly, e.g. by stimulating household robotics research. This paper is an attempt to identify a direct, commercially realistic application of sport robotics technology in the entertainment industry.

Our proposed system (cf. Fig. 1) is a minimalistic ball-playing robot serving at events, such as office parties, fairs or open house presentations. The robot is stationary and if a human throws a ball towards the robot it is supposed to hit it back, engaging the human in a robot-human ball game (at the moment it can technically only intercept not hit back). It is intended not as a long-term game, but rather as a short exciting experience being driven by the fascination

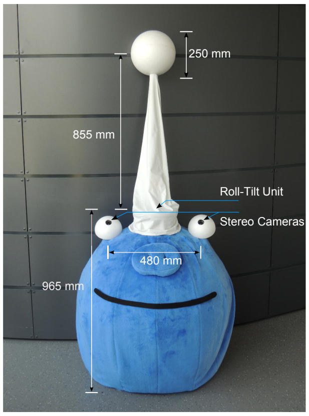


Fig. 1. The developed robot system, its major components, and dimensions. The computer and the motor's power supply are inside the robot's body.

of the unusual combination of sport, technology, and interactivity. Consequently, it is not an end-consumer product, rather than a device bought and operated by professional agencies organizing an event. The robot is specifically designed to meet the following requirements for this kind of operation:

1. *Safety.* For interacting with humans, the system has – of course – to be safe. It must be guaranteed that no person gets injured by the robot.
2. *Flexibility and easy appliance.* Similar to existing entertainment devices, the hardware and software must operate under various conditions. Amongst others, this concerns elements in the perceivable environment, lighting conditions, or the behavior of humans participating in the game. The system must be transportable and a non-expert must be able to set it up.
3. *Reasonably low costs.* The costs for the construction and maintenance of many current robot systems are not economical for any commercial activities. A successful entertainment robot should have a prize comparable to current non-robot entertainment devices.
4. *Reactivity.* A robot that interacts with people in a game (e. g. some kind of ball game) must have a level of reactivity comparable to that of humans. A significantly lower performance would result in a boring game that does not challenge the humans.
5. *Throughput.* Often events have many visitors and a large number of people should interact quickly with the system.

The contribution of this paper is the proposed design of a minimalistic ball-playing robot serving at events. We have developed a working, low-cost robot system that has been evaluated in various environments regarding different aspects. Although being complete at a technical level, the current module can only play a minimalistic game up to now. Thus it is a first step towards a commercial product.

This paper is organized as follows: Section 2 describes related work in the areas of sport robotics and tracking in sport environments. The developed event module and the underlying design considerations are presented in Sect. 3. Section 4 and Sect. 5 describe the necessary subcomponents for ball tracking and their calibration, followed by the motion control approach in Sect. 6. The robot's overall behavior and recent applications and experiments are presented in Sect. 7 and Sect. 8 respectively.

2 Related Work

Up to now, entertainment robots have been quite rare and are mostly settled in the context of artificial pets or toys. Prominent examples are the *Sony AIBO* [11] and the *RoboSapiens* [21] respectively. A few recent systems actually play ball games with humans. An example is Segway-soccer [2]. In this game, autonomous Segways play together and against humans standing on Segways. As a result, the actuation disadvantages of the robots are compensated. The same holds for the table soccer robot *KiRo* [22], where the competing humans only have access via rotary handles. This machine plays on a remarkable level and is even commercially available. Two robots that play simpler but more direct games are the *RoboKeeper* [9], a robot goalkeeper that parries penalty shots by professional soccer players, and DLR's *Rollin' Justin*, a humanoid torso that catches balls thrown by humans [5]. The former is already a successful product that can be hired for events, the latter is a research prototype actually built for service robotics not sports. Both systems share the same requirement: The ball has to be detected accurately and tracked robustly, so an exact prediction of upcoming trajectory is available in time and the robot can act accordingly.

Most of the previously presented work on perception systems with subsequent robotic actuation is based in the lab or controlled environments, usually extracting the ball due to its distinct color [18,19,3,9] or from the difference to a reference image [1,10]. A more flexible way would be to detect the ball as a circle in the image [24]. Due to the large measurement space, the native use of the popular Hough-Transform [14] is precluded. In [4], we proposed an efficient alternative that is also used here. Detected balls are usually tracked in 3-D either in a global or frame-by-frame based way. Simple global approaches fit a parabola to non-ambiguous ball measurements ([13,18,3]). Sophisticated global approaches also handling clutter exist (e.g. [23]), but their global operation forbids real-time usage. In contrast, frame-by-frame approaches such as Extended Kalman Filter (EKF) ([10,19]) are considered to be the optimal single-target tracking solution. They are generalized by Multiple Hypothesis Tracking [7] handling false-alarm measurement, starting and ending of ball flights in a sound probabilistic way [4].

Detecting and tracking balls is also of interest in broadcast television, with prominent examples for tennis and cricket [16], and for baseball [12]. The former is even used in the adjudication process after controversial calls by umpires, actually influencing the game's outcome.

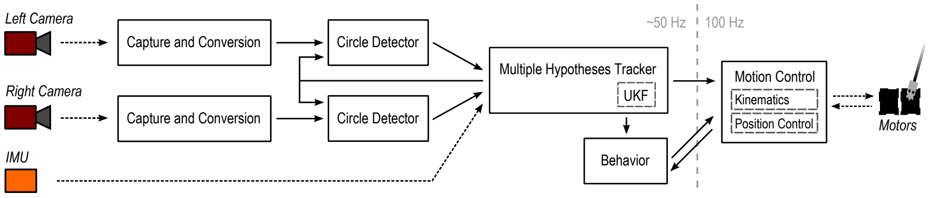


Fig. 2. The major software components of the presented robot system. Each box represents a single process (implemented as a ROS node). The arrows depict the data flow with solid arrows representing ROS communication links and dashed arrows representing direct communication with hardware devices.

3 System Overview

The contribution of the system presented in this paper is a combination of a minimalistic, low-cost hardware design and elaborated software which together provide the efficiency and performance needed for an interactive ball game.

3.1 Conception

The key idea in our robot design is to use as few degrees of freedom (DOF) as possible. The design of conventional 6 DOF robots is dominated by the fact that lower motors have to support and accelerate the weight of higher motors. Avoiding this setup dramatically reduces weight and costs and increases safety and reactivity.

How many DOF are needed to play a ball game? Surprisingly, two DOF suffice. First, we use a bat with spherical head to hit the ball, so we do not need to control the bat's orientation (cf. Fig. 1). This avoids three motors that need to be carried by other motors in conventional robots. Second, we attach the sphere with a rod to the first two robot axes leaving the elbow axis out. This reduces the robot's workspace to (roughly) a sphere, the missing DOF is contributed through the ball's motion by choosing when to hit the ball. With this design, the robot becomes inherently safe, as all moving parts are lightweight (335g) and soft with plastic foam for the sphere and a cushioned rod. For the same reason, we discarded mobility, because a mobile robot with battery, motors, and computing power would have a weight that makes inherent safety unrealistic at velocities needed for sports. A lower limit to the bat's weight comes from the fact that elasticity between motor and bat is needed to protect the motor from the ball's impact. Hence the impact mostly works as if bat and ball collide in free space, requiring the bat to be much heavier (230g) than the ball ($53-60$g).

An interactive game requires the robot to pass the ball back to a human player, raising the following question:

How much control has a 2 DOF robot over the ball? Surprisingly much: If the spherical bat hits the ball centrally, i. e. such that sphere and ball center would collide, the ball is reflected back into the direction it came from (apart from friction of course). With an offset in the two dimensions of the spherical workspace, the direction in which the ball is reflected can be controlled. Furthermore, the chosen bat velocity gives two additional DOFs on the reflected ball and also allows to add energy, if the ball is hit in motion. This leads to 4-DOF overall available to control the ball. Ball control is actually redundant as the motion of a ball reflected at a given point has only 3 DOF, namely velocity. We conclude, that two motors, at least in principle, suffice to intercept and control a flying ball in a minimalistic, hence cheap, and intrinsically safe setup.

Which kind of games is appropriate for such a robot? Realistically, with low-cost motors and intrinsic safety, the robot will not be competitive with a human player. In comparison, the RoboKeeper is competitive even with soccer professionals, but has a single expensive motor and needs a safety barrier and safety equipment. Our idea is to implement cooperative games that require passing between the players and the robot.

3.2 Hardware

The spherical head of the bat is made of foam-filled polystyrene, the rod is made of carbon and cushioned with rubber foam. Having a total height of 1.945m (cf. Fig. 1), the robot's workspace roughly resembles the one of a child stretching its arms. The bat is moved by two *Dynamixel EX-106+* servo motors that are paired to a roll tilt unit. These motors provide nominally 10.49Nm drive torque, well above the 2.9Nm needed to hold the bat. The nominal speed without load is 546°/s, effectively we operate at $180°/s$ with 70% torque to reduce gear load. This still allows sufficiently dynamic actuation (cf. Sect. 8). We use different, unmodified toy rubber balls ($53-60$g weight, $90-121$mm diameter) that are small and light enough to be played by the robot and heavy enough to be conveniently thrown by a human.

The robot's main sensors are two synchronized *AVT Marlin F-046* FireWire cameras. To be able to track a ball during its whole flight (cf. Fig. 3), the cameras are equipped with lenses of 4.8mm focal length ($\approx 57°$ horizontal field-of-view) and are mounted on the robot in an angle of about 35°. We have also tried Microsoft's Kinect sensor, but found that it cannot detect the ball due to motion blur. Furthermore, an IMU is used to provide gravity information to the ball flight tracker as well as to determine impacts on the robot. All computations are performed by a modern personal computer (*Intel Core i7 860* 4×2.80GHz). The overall hardware costs of the robot as presented in this paper are less than 5000€ (including: 1000€ computer, 2500€ sensors, 1000€ actuators), fitting the requirement of reasonable low costs. The whole system is self-contained, i. e. it does not require any further constructions or modifications in the environment, such as external sensors or special lighting.

The current robot hull is designed as a plushy blue pig to provide a pleasant and funny counterpart to human players. However, the system is not technically bound to any specific character and might have a completely different appearance in the future.

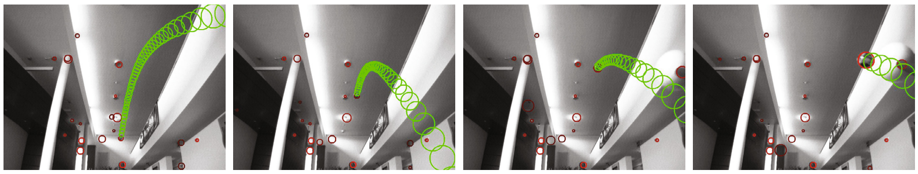


Fig. 3. A ball tracked indoors. Circles (red) are detected in the image by a contrast-normalized gradient criterion and passed to a Multi Hypothesis Tracker (MHT) that handles clutter and missing detections and uses an Unscented Kalman Filter (UKF) to estimate position and velocity that are used to predict future states (green circles).

3.3 Software

The software consists of a number of components depicted in Fig. 2. The whole system is embedded in the Robot Operating System (ROS) [17] framework, running on Ubuntu Linux. Each component is a separate process, so the computationally demanding image processing and ball tracking components utilize the available CPU cores. The overall architecture is quite straightforward: Images (780×580 8 Bit grayscale) are captured with about 50 Hz. On these images, a generic search for circles is performed. The circles from both camera images are fused within the tracking component (cf. Sect. 4). The most likely ball tracks are sent back to the circle detection processes, allowing a refined search for circles around the predicted position. Finally, the robot's motors are controlled to steer the bat to the first intersection point of the ball's trajectory with the robot's workspace (cf. Sect. 6). In case of no tracks, some other actions are carried out by the robot (cf. Sect. 7). Position control runs with 100 Hz on the PC, as the servo's built-in controller is not able to handle the large inertia in our setup.

4 Ball Tracking

The subcomponent for tracking balls is the two-staged bottom-up approach introduced in [4], which is also in use for DLR's ball-catching humanoid *Rollin' Justin* [5]. In the first stage, balls are detected as circles (cf. Fig. 3). By computing the average fraction of image energy that is a radial gradient and evaluating this gradient response along the contour of all possible circle positions and radii, the N best circles in the image are determined. To achieve real-time operation, evaluation of all circle responses is done in a multiple-scale fashion. The detector is illumination invariant. Also, in contrast to other ball detection methods that detect the ball by its color ([3,6,18,19]), this method needs no calibration, facilitating robust and easy appliance.

In the second stage, the detected circles are passed to a multi-target filter, namely the Multiple Hypothesis Tracker (MHT) [7], to estimate the cardinality and the individual states, i. e. position and velocity, of all flying balls. MHT seeks for the data association hypothesis with the highest posterior probability. On arrival of a new set of measurements, each existing hypothesis from the previous time step is expanded to a set of new hypotheses by considering all possible

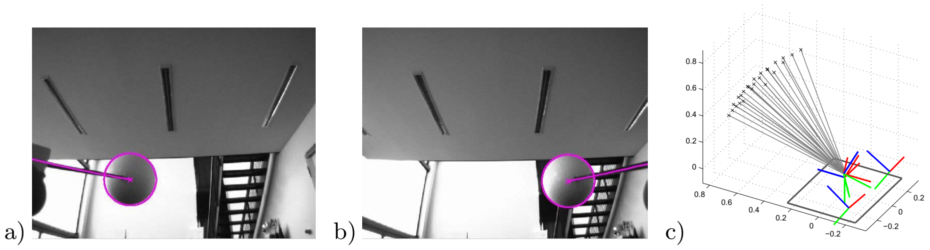


Fig. 4. a) and b) Left and right camera image showing the head of the bat and its projection as a circle and a line obtained from the calibration given the measured joint angles. c) Positions of the head's center (marked as a cross) and of the bat (line) used for calibration and visualization of all frames involved in the calibration procedure.

associations of measurements to tracks. Also, handling of track initiation and termination, as well as the case of having clutter measurements, is considered while generating hypotheses. To increase robustness, prior information encoding typical positions from where the ball is thrown and that the ball is thrown towards the robot is used when starting tracks. This helps generating only the tracks actually demanding the robot to react and discards any that might accidentally look like possibly flying ball (e. g. a moving head). The states of the involved tracks are estimated using an Unscented Kalman Filter (UKF) [20].

5 Calibration

Although being mechanically simple, a calibration of the setup has to be performed prior to playing. Fortunately, sensors and actuators can be *factory calibrated*, since all components are rigidly mounted and not expected to change.

During the robot's normal operation, the tracks estimated by the MHT must be converted from the camera frame into the base frame of the kinematic chain. We calibrate this transformation, the joint's scale factors and offsets, the camera's intrinsic parameters and stereo calibration jointly by using a checkerboard and by observing how the spherical head of the bat moves in both cameras when joint angles change (cf. Fig. 4). Given both joint angles $\theta_{roll}, \theta_{tilt}$, the forward kinematics, transforming from the frame of the tilting servo into the base frame, reads as a chain of transformations

$$M_{\theta_{roll}, \theta_{tilt}} = M(\theta_{roll}) \cdot T \cdot M(\theta_{tilt}) \tag{1}$$

with

$$M(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, T = \begin{bmatrix} \cos\frac{\pi}{2} & 0 & \sin\frac{\pi}{2} & 0 \\ 0 & 1 & 0 & l_{off} \\ -\sin\frac{\pi}{2} & 0 & \cos\frac{\pi}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

$M(\theta)$ represents a rotation along the joint axis and T describes the fixed rotational and translational displacement between the joints. Here, l_{off} is -35 mm.

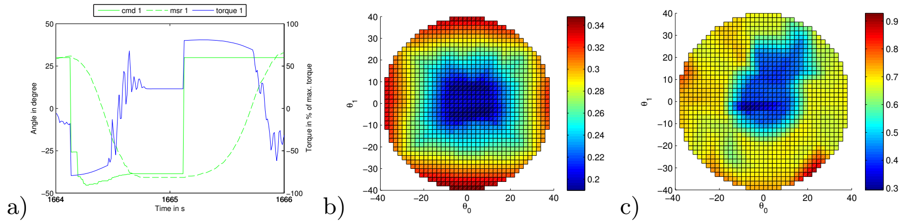


Fig. 5. a) Position controller's behavior: commanded (green) and measured (dashed green) angle and motor torque (blue). **b)** Required time [s] to reach a joint angle configuration (θ_0, θ_1) when starting from $(\theta_0 = 0, \theta_1 = 0)$ with $\leq 2°$ error. **c)** Required time [s] to reach $\leq 0.5°$ error. The robot reached the whole workspace reasonably quickly (< 0.34s), but needs significant time (> 0.4s) to damp oscillations. This problem is mainly caused by backlash and has to be addressed in the future. The steady state error was always $< 0.4°$. Overall, the system appears fast enough for a ball game.

Calibration data is acquired in two consecutive steps. First, observations of a checkerboard from different views define camera parameters and their relation (stereo). In the second step, a grid of different bat poses (cf. Fig. 4, right) is commanded and corresponding joint angles and camera images are saved. Measuring the spherical head as a circle and the connection to the tilt frame as a point on the rod, the transformation between the kinematic base frame and the cameras as well as the servo's scale factors and offsets are defined.

From 9 checkerboard images and 25 bat poses, this procedure results in a residual rms $(0.62, 0.88)$ px for checkerboard corner positions and $(3.8120, 2.2419)$ px and 0.7307 px for head positions and radii, respectively.

6 Motion Control

The motion control component has the task to move the bat to a location where it hits the ball in midair. Precisely, the trajectory of the ball's center should hit the bat's center. Hitting the ball non-centrally or in motion, as outlined in Sect. 3.1, is future work. The approach is straightforward: First, angles for both servo motors are computed by inverse kinematics. Afterwards, the movement is executed by a textbook PD controller.

The predicted ball tracks from the MHT are converted into robot-base coordinates and the one closest to a position reachable by the robot is determined. For this task, an analytical inverse of (1) is used:

$$\theta_{roll} = \arctan \frac{x}{z} \quad \text{for } z > 0 \tag{2}$$

$$\theta_{tilt} = -\arctan \frac{y}{\sqrt{x^2 + z^2} - l_{off}} \tag{3}$$

where x, y, z are in robot base coordinates and l_{off} is the displacement between the two servo axes. Feeding the calculated angles back into (1) allows to determine the distance to a reachable position and therefore the closest position where the ball is intercepted. The resulting angles are passed to the position controller that actuates the servo motors based on the commanded and measured servo angles. In addition, it enforces position limits to avoid hardware damage. The core of this component is a PD-Controller for each servo with angle-position error e_t as process variable and percentage of maximum torque as output.

$$\tau_{out} = k_p e_t + k_d d_t + \tau_{comp} \tag{4}$$

where d_t are the derivatives of e_t, low-pass filtered by

$$d_t = e^{-2\pi f \Delta t} \cdot d_{t-1} + (1 - e^{-2\pi f \Delta t}) \cdot \frac{e_t - e_{t-1}}{\Delta t} \tag{5}$$

with f as the cutoff frequency defined later. To compensate the steady-state errors induced by the bat's weight, a torque τ_{comp} is added, assuming a weight F and the center of gravity being in the sphere center.

$$\tau_{comp_{roll}} = -(l \cos \theta_{tilt} + l_{off}) F \sin \theta_{roll} \tag{6}$$

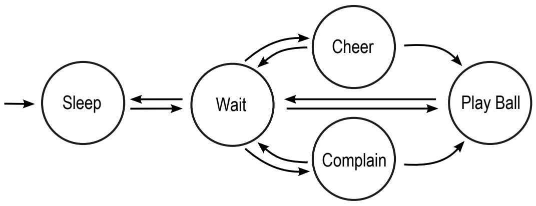
$$\tau_{comp_{tilt}} = -lF \cos \theta_{roll} \sin \theta_{tilt} \tag{7}$$

The controller has to cope with two main sources of inaccuracy. The first one are elasticities of the outer hardware, i. e. the post and connectors which allow oscillation of the end effector on acceleration. The second is large backlash (ca. 3°), presumably inside the servo gear. This is a common problem of low-cost servos. It causes, together with latencies of the control loop, massive oscillations of ca. 24Hz if k_d is increased. To reduce this effect and allow for higher k_d and k_p the cutoff frequency of the low-pass filter on the derivations is chosen as f = 20Hz. The controller's typical operation is shown in Fig. 5a.

7 Robot Behaviors

The sole ability to return balls is – independent of its quality – not satisfying for a robot that is made to entertain people. Therefore, a couple of interactive behaviors have been added to the system to give users some kind of appealing feedback as well as the impression of an intelligent robot. Even with only two 2 DOF, some meaningful gestures are already possible.

The overall behavior is realized as a simple finite state machine (cf. Fig. 6). After booting the system, the robot remains in the *sleeping* state until its bat has been lifted by a user. This haptic state transition allows an operation without any additional external controls. In its *waiting state*, the robot does not perform any actions and awaits a flying ball. If no ball is thrown for a certain amount of time, the robot starts to *cheer* the user by moving its bat and playing some sound. A similar gesture is performed in case of *complaining* about a badly thrown ball that did not reach the robot's workspace. In case of a thrown ball, the cheering and complaining states can both be interrupted at any time to avoid any mode confusion, i. e. to avoid the user's impression that the robot ignores the ball. In addition to these states, two simple reactive behaviors, which are

a)


b)

Fig. 6. a) The finite-state machine that controls the robot's behavior. b) Demonstration at the DFKI's booth during the CeBIT 2012. Visitors throw balls towards the robot.

always active, have been added. Whenever the robot is unexpectedly hit by a ball or pushed by a person, the event can be detected by the built-in IMU and a loud "Ouch!" sound is triggered. Furthermore, the bat's waiting pose is dynamically adjusted to always face the player, providing an impression of the robot's attention. The according person tracking is currently realized using a *Microsoft Kinect* sensor. To save this additional sensor and to be able to track persons outdoors, an alternative image-based solution using the *Histogram of Oriented Gradients* approach [8] is currently under development. Most of these behaviors are shown in the supplementary video[1].

8 Experimental Results and Applications

In a first experiment, the reachability of the workspace given a limited amount of time was determined. The results that are depicted in Fig. 5 indicate a sufficient speed and reactivity for an interactive game.

Given a technically working system, the most obvious experiment is to throw balls towards the robot and count how often it really plays the ball. We conducted this inside the entrance hall of our research building as well as on the lawn next to the building. The latter experiment was conducted during the afternoon, having a sunny sky with several clouds. In both experiments, the ball was thrown from about 5-6 meters which we consider as a reasonable distance in a real game. Different persons have thrown balls. As presented in detail in Tab. 1, the robot was able to play 84% of all balls that reached its workspace, even 95.1% outdoors. However, there is a remarkable difference why the robot fails to play the ball if it does. Indoors, the major problem was ball tracking, 14.6% of the balls crossing the robots workspace have not been tracked, outdoors only 4.9%. At a first glance, this is surprising since computer vision under natural lighting conditions is considered to be more error-prone. In fact, the applied approach is linearly illumination independent but affected by circular clutter. The latter occurs indoors (cf. Fig. 3) but not in the sky. Instead, the major

[1] `http://www.informatik.uni-bremen.de/ timlaue/video/piggy_demonstration.mp4`

Table 1. Results of the conducted hitting experiments

	Throws	Hits total	Workspace missed	Hits in Workspace	Not tracked	Missed
Indoor	196	145 (74.0%)	18 (9.2%)	81.5%	14.6%	3.9%
Outdoor	56	39 (69.6%)	15 (26.8%)	95.1%	4.9%	0%
Total	252	184 (73.0%)	33 (13.1%)	84.0%	12.8%	3.2%

problem outdoors was the throwers missing the robot's workspace, probably caused by wind. In both experiments, the robot nearly always hit a tracked ball.

The system has already been presented to the public on several occasions. One particular highlight was the CeBIT 2012 computer fair in Hanover, Germany, at which the robot has continuously played with arbitrary visitors for five whole days, nine hours a day. At this demonstration, the robot proved to be already applicable under realistic conditions (cf. Fig. 6). In particular, it is well suited for active entertainment of a large number of people: We simply continuously handed balls to the crowd who threw them towards the robot without any instructions necessary.

Regarding a commercial perspective, discussions with an entertainment professional were very positive: Even when assuming a final robot price of €15000, written-off in 5 years, plus €1000/year maintenance, yearly costs would be €4000. With a realistic price of €950 per use minus 2×€200 personnel costs and 30 uses per year, we would have €12500 (or 83%) return of invest per year.

9 Conclusions and Future Work

In this paper, we presented a commercially realistic entertainment robotics application beyond a lab demo scenario. The current version of the proposed robot system already fulfills important requirements for an interactive entertainment robot, such as safety, low costs, and flexibility regarding setup and application. In different experiments, we demonstrated the reactivity needed for playing a minimalistic first ball game. However, for more sophisticated ball games, preferably including multiple persons, several issues still have to be addressed in the future. To allow tracking in a larger area around the robot, the cameras need to be moved. This could be realized almost cost-neutral by replacing the robot's roll tilt unit by a pan tilt construction and mounting the cameras above the pan joint. Furthermore, the precision of the system needs to be increased to achieve multi-directional reflections reliably.

References

1. Andersson, R.L.: Dynamic sensing in a ping-pong playing robot. IEEE Trans.on Robotics and Automation 5(6) (1989)
2. Argall, B., Browning, B., Gu, Y., Veloso, M.: The first segway soccer experience: Towards peer-to-peer human-robot teams. In: Proc. Conf. on Human-Robot Interaction (2006)
3. Bätz, G., Yaqub, A., Wu, H., Kühnlenz, K., Wollherr, D., Buss, M.: Dynamic manipulation: Nonprehensile ball catching. In: Proc. IEEE Mediterranean Conf. on Control and Automation (2010)

4. Birbach, O., Frese, U.: A multiple hypothesis approach for a ball tracking system. In: Fritz, M., Schiele, B., Piater, J.H. (eds.) ICVS 2009. LNCS, vol. 5815, pp. 435–444. Springer, Heidelberg (2009)
5. Birbach, O., Frese, U., Bäuml, B.: Realtime perception for catching a flying ball with a mobile humanoid. In: Proc. IEEE Int. Conf. on Robotics and Automation (2011)
6. Bruce, J., Balch, T., Veloso, M.: Fast and inexpensive color image segmentation for interactive robots. In: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 2061–2066 (2000)
7. Cox, I.J., Hingorani, S.L.: An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. IEEE Trans. on Pattern Analysis and Machine Intelligence 18(2), 138–150 (1996)
8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proceedings of the 2005 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, pp. 886–893 (2005)
9. Fraunhofer, I.M.L.: 4attention GmbH & Co. KG: Robokeeper web site (2011), `http://www.robokeeper.com`
10. Frese, U., Bäuml, B., Haidacher, S., Schreiber, G., Schaefer, I., Hähnle, M., Hirzinger, G.: Off-the-shelf vision for a robotic ball catcher. In: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (2001)
11. Fujita, M., Kitano, H.: Development of an Autonomous Quadruped Robot for Robot Entertainment. Autonomous Robots 5(1), 7–18 (1998)
12. Guéziec, A.: Tracking pitches for broadcast television. Computer 35(3), 38–43 (2002)
13. Hove, B., Slotine, J.: Experiments in robotic catching. In: Proc. of the American Control Conf., pp. 380–385 (1991)
14. Kimme, C., Ballard, D., Sklansky, J.: Finding circles by an array of accumulators. Comm. of the ACM 18(2) (1975)
15. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., Matsubara, H.: RoboCup: A challenge problem for AI. AI Magazine 18(1), 73–85 (1997)
16. Owens, N., Harris, C., Stennett, C.: Hawk-eye tennis system. In: Int. Conf. on Visual Information Engineering, pp. 182–185 (2003)
17. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: Proceedings of the Open-Source Software workshop of the International Conference on Robotics and Automation (ICRA), Kobe, Japan (2009)
18. Riley, M., Atkeson, C.G.: Robot catching: Towards engaging human-humanoid interaction. Autonomous Robots 12(1), 119–128 (2002)
19. Smith, C., Christensen, H.I.: Using COTS to construct a high performance robot arm. In: Proc. IEEE Intern. Conf. on Robotics and Automation (2007)
20. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
21. Tilden, M.W.: Neuromorphic robot humanoid to step into the market. The Neuromorphic Engineer 1(1), 12 (2004)
22. Weigel, T., Nebel, B.: KiRo - an autonomous table soccer player. In: Kaminka, G.A., Lima, P.U., Rojas, R. (eds.) RoboCup 2002. LNCS (LNAI), vol. 2752, pp. 384–392. Springer, Heidelberg (2003)
23. Yan, F., Kostin, A., Christmas, W., Kittler, J.: A novel data association algorithm for object tracking in clutter with application to tennis video analysis. In: Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition (2006)
24. Yuen, H., Princen, J., Illingworth, J., Kittler, J.: A comparative study of hough transform methods for circle finding. In: Alvey Vision Conf., pp. 169–174 (1989)