2012 12th IEEE-RAS International Conference on Humanoid Robots
Nov.29-Dec.1, 2012. Business Innovation Center Osaka, Japan

Playing Catch and Juggling with a Humanoid Robot

Jens Kober*†, Matthew Glisson*, and Michael Mistry*‡
*Disney Research, Pittsburgh, 4720 Forbes Ave., Ste. 110, Pittsburgh, PA 15213, USA
†Bielefeld University, Germany
‡University of Birmingham, UK
jkober@cor-lab.uni-bielefeld.de, matt.glisson@disneyresearch.com, m.n.mistry@bham.ac.uk

***Abstract*—Entertainment robots in theme park environments typically do not allow for physical interaction and contact with guests. However, catching and throwing back objects is one form of physical engagement that still maintains a safe distance between the robot and participants. Using a theme park type animatronic humanoid robot, we developed a test bed for a throwing and catching game scenario. We use an external camera system (ASUS Xtion PRO LIVE) to locate balls and a Kalman filter to predict ball destination and timing. The robot's hand and joint-space are calibrated to the vision coordinate system using a least-squares technique, such that the hand can be positioned to the predicted location. Successful catches are thrown back two and a half meters forward to the participant, and missed catches are detected to trigger suitable animations that indicate failure. Human to robot partner juggling (three ball cascade pattern, one hand for each partner) is also achieved by speeding up the catching/throwing cycle. We tested the throwing/catching system on six participants (one child and five adults, including one elderly), and the juggling system on three skilled jugglers.**

I. Introduction

For nearly five decades, animatronic robots have provided entertainment to theme park guests by reproducing natural human or animal motion. These robots add to storytelling domains a sense of physical presence and realism not possible by graphics or video. Most animatronic systems in place today, however, offer little in terms of interaction. The robots in theme parks generally play back pre-scripted motion, with no possibility of adaptation or reaction to the environment or guests. In order to increase levels of both realism and story engagement, we want to be able to establish a *physical* connection between animatronic characters and theme park guests, and change the guest's role in the story from mere observer to participant. Throwing and catching objects from guests to robots is one way of accomplishing such interaction, while still maintaining a safe distance. With this objective in mind, we undertook the development of a test bed for real-world human-robot physical interaction by means of throwing and catching balls between a human and humanoid robot.

Taking the scenario of "playing catch" with a robot to a further level, we also examined human to robot partner juggling. Here ball exchange must continually take place with at least one ball always in the air. While robots have been made to juggle before, we are particularly interested in the tight and highly dynamic interaction required between the human and robot. A robot that can juggle by itself or with other robots can be very impressive, but merely showcases the repeatability and precision expected in robotic motion. Juggling with a human partner demonstrates an ability to react and respond to human inconsistency, potentially resulting in more life-like motion and an increased sense of realism.

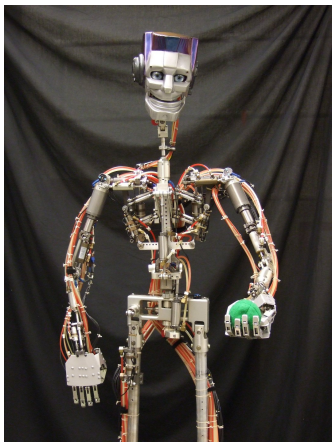

Fig. 1. Sky, a Disney A100 Audio-Animatronics hydraulic humanoid robot

Robots capable of catching and throwing have been investigated previously. One of the first ball catching robots applications was based on fitting a parabola to the vision data and matching the robot trajectory to the ball trajectory [1]. Catching a ball with a humanoid wearing a baseball glove has been studied by Riley and Atkeson [2]. The movements were generated using movement primitives. Catching balls is also possible without reconstructing the 3D ball position by visual servoing [3], [4]. A robot tossing a cellphone and catching it again has been studied at the University of Tokyo [5]. The approach is based on high-speed sensory-motor fusion. Nonprehensile Ball Catching has been studied in [6] where the ball trajectory is predicted using recursive least squares. Catching a ball thrown towards a robot has been achieved by planning [7] and supervised learning [8] at DLR. The corresponding perception system are based on background subtraction [9] or circle detectors [10], [11] and Kalman Filters. The complete DLR system is discussed in [12].

Robot juggling has been studied by Aboaf et al. [13], [14]. The juggling task here corresponds to hitting a ball with a

978-1-4673-1369-8/12/$31.00 ©2012 IEEE

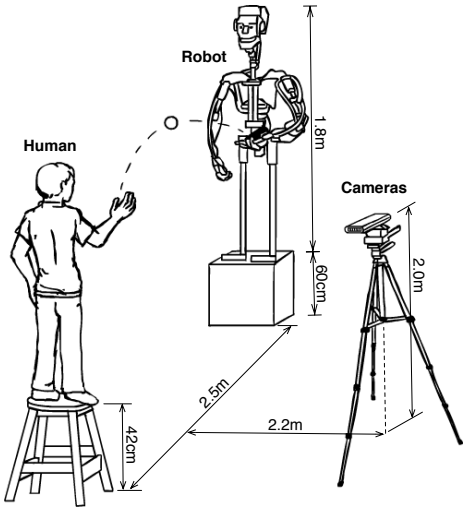

Fig. 2. Graphical overview of our experimental setup for catching and throwing with a humanoid robot. The participant stands on a stool to be at equal level with the robot. A ASUS Xtion PRO LIVE camera system acquires the 3D position of the ball. Both the camera and the robot can only be used indoors.

paddle. Rizzi et al. [15], Yu et al. [16], as well as Schaal et al. [17]–[19] studied similar tasks. Juggling balls with a single arm and a funnel shaped hand has been studied by Sakaguchi et al. [20], [21]. Atkeson et al. studied juggling three balls in a cascade pattern with a Sarcos humaoid robot [22]. We believe we are the first to demonstrate human to robot partner juggling.

II. System Description

An overview of our experimental setup for throwing and catching is shown in Fig. 2. A description of the robot hardware is in Section II-A. The vision system is discussed in Sections II-B and II-C, and the robot software in Sections II-D through II-G.

A. Robot Hardware

We use Sky, a Walt Disney Imagineering A100 Audio-Animatronics figure (Fig. 1) with 39 degrees-of-freedom, 38 of which are driven by hydraulic actuators. This type of robot platform is currently commonly employed in the theme parks. The robot stands on a 60cm base containing its hydraulic valve manifold, pressure transducers, and computer connections. Its feet are fixed to the base so stability and balance are not a concern. For throwing and catching we use the left arm, which has seven degrees of freedom plus five fingers with one degree of freedom each. For additional motions such as orienting the robot towards the participant, simulating ball following, and acknowledging catching failure, we additionally use the pelvis, torso, shoulder shrugs, neck and eyes. The control system allows for the update of desired joint position set-points at 30Hz, a rate considered quite slow for reactive control. However, as this robot was designed only for pre-recorded trajectory playback, it is a limitation we must cope with. Lower level hydraulic valve controllers realize the desired positions of each actuator at a control loop of 1kHz. The maximal hand velocities is approximately 1.5m/s. We augmented the left hand of the robot with a plate to cover and protect the finger actuators and a foam rim to provide a more cup-like shape suitable for catching (Fig. 3). Our goal was to maintain as much of a human-like appearance as possible.

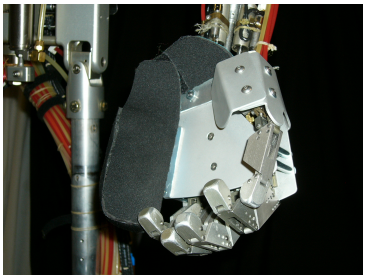

Fig. 3. Detail of the robot's catching hand. The palm of the hand is approximately 10cm square.

B. Vision System

As the robot has no onboard cameras in its eyes or elsewhere, we use an external camera system to acquire the 3D position of the balls. We employ a low cost off-the-shelf ASUS Xtion PRO LIVE, which contains both a depth and color camera, and does not require internal and external calibration of a stereo camera setup. This camera is almost identical to the Microsoft Kinect but supports hardware synchronization of the color and depth stream. The camera runs at 30Hz, the same as the control rate of the robot, thus avoiding aliasing. Ball detection is done with OpenCV, which obtains the registered color and depth images from the Xtion via OpenNI. The color image is a standard 640x480 8bit RGB image.

An overview of our vision pipeline is shown in Fig. 4. The first step is to remove the foreground and the background by creating a mask based on a thresholded depth image. The color image is then masked and converted to HSV space. HSV is a convenient space to threshold on color while being invariant to illumination. Small false positives are removed by morphological operations. We run a Hough circle detector on the resulting image, which rejects larger false positives and yields the centers of the balls in the image plane in pixels.

We average the depth value of the pixels that are active both in the color and the depth mask to obtain the depth of the balls[1]. From the pixel position, focal length, and depth value, the complete Cartesian position can be reconstructed. The whole processing takes approximately 30ms.

Using the same camera, but a separate processing pipeline, we additionally track the user's location via the OpenNI skeleton tracker. From the skeleton information, we can isolate the 3D position of the user's head, which we use to orient the robot towards the user and set its gaze to be looking at the participant.

[1]The resulting offset in camera direction is negligible and would be compensated by the data-driven calibration.

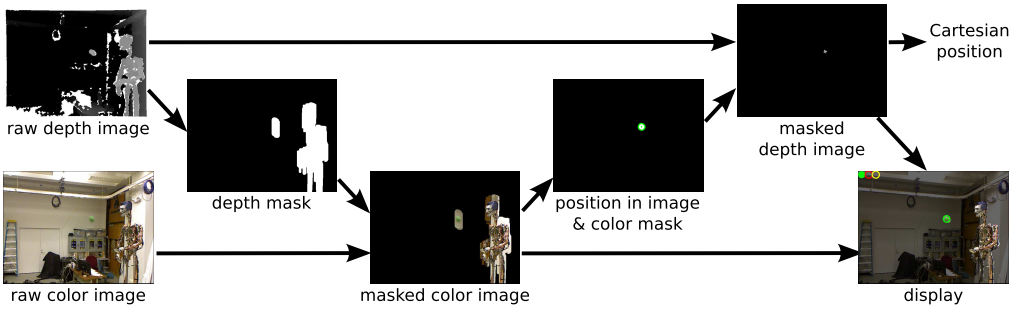

Fig. 4. Overview of the processing steps in the vision system.

C. Ball Filtering and Prediction

The vision system obtains the Cartesian positions of the balls. In order to predict the catching position for the hand, we also need ball velocities. As the distance is short and the balls are relatively heavy (approx. ø 7cm, 100g), modeling the air drag did not improve the prediction and the balls can be modeled as point masses, i.e.:

$$\mathbf{x}(t) = \mathbf{x}_0 + \mathbf{v}_0 t + \mathbf{a}_0 t^2, \tag{1}$$

where t is the time, $\mathbf{x}_0$ the initial position, $\mathbf{v}_0$ the initial velocity, and $\mathbf{a}_0$ the initial acceleration. In order to predict the ball trajectory, we need to estimate the initial values. We evaluated keeping the initial acceleration fixed to $\mathbf{a}_0 = [0\ 0\ -9.81]\, m/s^2$ but found that the prediction is better if we keep this parameter open as the camera may not be precisely aligned to gravity. Using all previously seen ball positions and estimating the parameters in a least squares sense, required too many samples to converge and the robot did not manage to catch reliably as the initial predictions were too far off. We use a linear Kalman filter instead that is re-initialized every time a new ball is thrown. The initialization values are obtained by finite difference. A standard deviation of 5cm and 5mm for the measurement and process noise respectively worked well in practice.

The robot always attempts to catch the ball on a predefined horizontal plane. Using Eq. (1), the time at which the ball trajectory intersects this plane is calculated. Then the corresponding intersection position can be determined.

D. Robot State Machine

For our catching game, the robot continually waits for a ball to catch, and throws it back soon after catching. We implement this behavior as a state machine (Fig. 5). Catching (Section II-E) and throwing (Section II-F) are two distinct states and we additionally include two states to handle the smooth transitions in between catching and throwing. If the robot does not catch the ball, we transit to a missed ball state (Section II-G2) and return to the waiting position for a new ball. Switching from the catching state to the transition or the missed ball state is tied to the predicted catching time and the ball position. All other transitions occur after a fixed amount of time has elapsed.

E. Catching Algorithm

The robot receives the predicted catching position as well as the predicted time until the catch from the vision system. If the predicted catching position is within the robot's catching region, the hand is moved to the new position. Due to the low control frequency, we employ filtered step functions as desired trajectories for this task. At the time of predicted catch the fingers are closed as much as possible to trap the ball, however, the fingers are not capable of fully enclosing the ball and preventing it from escape. Other than our vision system, we have no means to detect success of a caught ball (e.g. no contact sensor or switch on the palm). Because the vision system sometimes sees the ball in the hand, it can lead to strange behaviors if the robot continues to try to catch the ball even if it is already caught. Consequently we stop updating the predicted catching position and catch time if there is less than 66ms until catch (two control cycles). This cutoff also prevents very fast movements when the ball is landing in the hand, which sometimes led to the robot hitting the ball and having it bounce off.

1) Inverse Kinematics Algorithm: Inverse kinematics is required to map the desired hand catching position to joint commands. Most inverse kinematics approaches work in the reference frame of the robot. Thus we would have

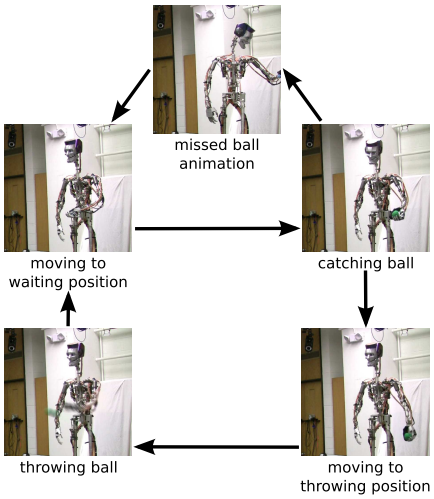

Fig. 5. The robot's state machine and transition diagram. Photos indicate the end positions of each state.

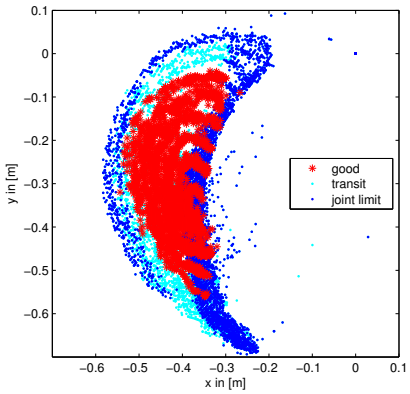

Fig. 6. This figure illustrates top-down view of the feasible catching region of the robot. The robot stands at position (0,0) and is facing to the left. The region is determined by the workspace of the hand before hitting a joint limit (blue). The linear inverse kinematics algorithm performs well (position error less than size of catching hand) in almost the whole region (red). IK error increases in the cyan region but the robot manages nevertheless to catch reliably in the lower left-hand region, see Fig. 7. The average error in the red area is 3cm and 15cm in the cyan area. Slight overlap between red and blue regions is due to joint limits effectively forcing the hand to move out of the sampling plane.

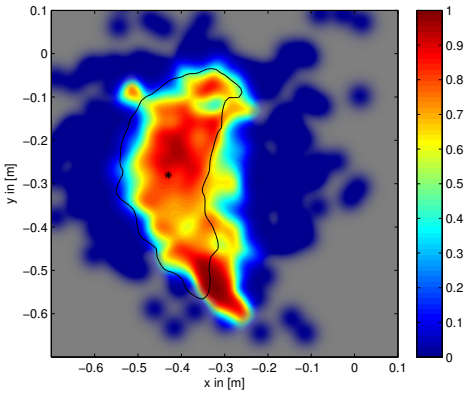

Fig. 7. This figure illustrates the catching performance of the robot. The view is identical to Fig. 6. The color-scale indicates the success rate at a given location. The black outline marks the feasible catching region (red region in Fig. 6) and the black star indicates the default hand position. The figure shows average results for 200 throws by novice users and 400 throws by expert users. Within the feasible catching region the success rate is 75%. The data was smoothed and normalized regionally to produce this plot. Due to the shape of the hand and having balls always fly predominantly in a positive x-direction, the robot also performs well catching balls to the right and below the feasible catching region.

to additionally calibrate the robot's reference frame to the camera's. Instead we decided to use a single local linear model that combines both the vision-to-robot calibration and the inverse kinematics. Because catching always occurs on the same horizontal plane in a fairly limited region, the mapping from Cartesian positions to joint angles can be approximated as linear. The position θ_i of joint i is expressed as a function of the Cartesian position (x, y, z) in the following form:

$$\theta_i\left(x, y, z\right) = \alpha_{i,x}x + \alpha_{i,y}y + \alpha_{i,y}z + \alpha_{i,o},$$

where $\alpha_{i,\{x,y,z\}}$ are constants corresponding to the Cartesian positions and $\alpha_{i,o}$ is a constant offset.

The parameters $\boldsymbol{\alpha}_i = [\alpha_{i,x}\ \alpha_{i,y}\ \alpha_{i,z}\ \alpha_{i,o}]$ are fitted using linear regression (least squares). We attach a ball to the robot's open hand and run a predetermined calibration trajectory that moves the hand to span the expected catching area. The joint angles and the corresponding Cartesian positions are stored. At the end of the trajectory the parameters for each degree of freedom i are calculated

$$\boldsymbol{\alpha}_i = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{\Theta}_i,$$

where the rows of $\mathbf{X}$ contain $[x_t\ y_t\ z_t\ 1]$, i.e., the Cartesian positions at time-step t and an offset, $\boldsymbol{\Theta}_i$ is a vector with the corresponding joint positions, $\mathbf{I}$ is the identity matrix, and λ is a small ridge factor. This type of approximated inverse kinematics works sufficiently well in the catching region of the robot. We tested the inverse kinematics in the work-space of the robot by executing a reference trajectory and comparing the desired hand position to the positions measured by the vision system. We picked the height with maximum coverage as the horizontal catching plane. Fig. 6 illustrates the region the robot's hand can cover without hitting joint-limits and where the error in the inverse kinematics approximation is less than the variance compensated by the size of the hand (<5cm).

The data-driven approach allows for quickly modifying the setup, e.g., by moving the camera. At the same time, the learned model automatically compensates inaccuracies in the camera's depth perception. Additionally, the A100 figures are highly modular and their kinematics can be extensively reconfigured. After such a change, only the calibration procedure needs to be run again in order to estimate new parameters. The linear model also provides a unique mapping from Cartesian positions to joint angles and does not require any iterative steps.

F. Throwing Approach

After a successful catch, the robot will transition to throwing back the ball. The arm moves to a predefined position from which the throw starts. To throw, the robot moves its hand on a forward arc using the shoulder, elbow and wrist joints. At the same time the fingers open to release the ball. As the maximum velocity of the robot is too limited to get a good throw only by releasing the ball, we need to stop the movement as abruptly as possible to make the ball detach from the hand. We use a step-function that accelerates the robot to its maximum velocity and then brakes abruptly. Currently we only have a single fixed throwing movement. The maximum distance we could achieve was approx. 2.5m. Possible improvements include making the throw dependent on how far away the user stands.

G. Interactive Cues and Animation

We found that adding subtle motions in addition to the functionality required for catching or throwing, generally made the system more appealing. For example, an early version of the system could not detect whether the ball had been caught successfully or not. Users appeared irritated

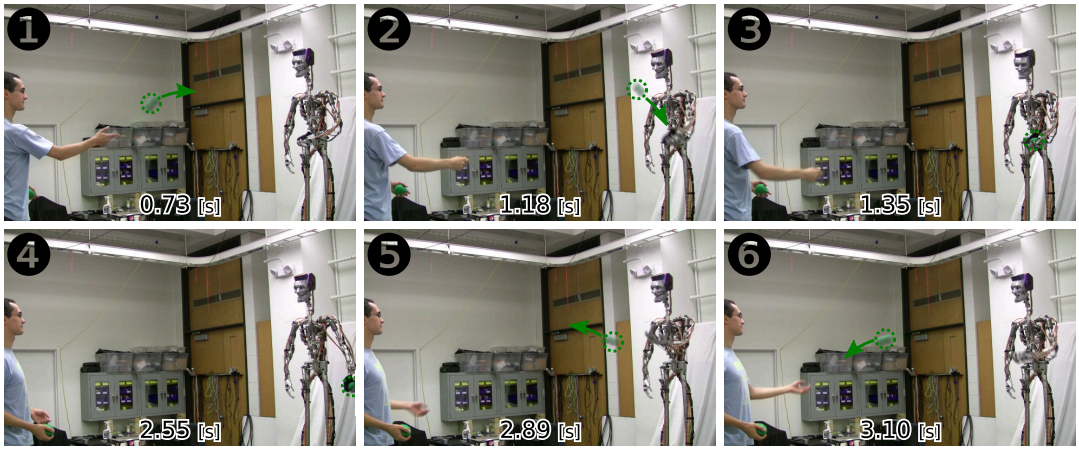

Fig. 8. An example sequence of playing catch. Green circles highlight the location of the ball. The participant throws the ball to the robot, and after catching, the robot tosses the ball back.

when the robot did not react appropriately to a missed ball and attempted to throw back empty handed. We therefore use the vision system to detect ball catching failures, and acknowledge them with an appropriate animation. Additionally, head motions were added to give the appearance of ball and participant tracking, as opposed to unnaturally staring out into space.

1) Head and Body Orientation: As discussed in [23], a realistic gaze behavior is essential to render catching animations believable. The robot appears to track the ball with its head and eyes while the ball is thrown towards it and it is attempting to catch. At the time of throw initiation, we reset the head and eye position such that it always looks directly at the user. The head motion was subtle, but generally added to the robot's appeal. While waiting for a new ball, the gaze tracks the user and additionally the body turns to face the user.

The user tracking is based on the OpenNI skeleton tracker. In order to achieve a natural looking movement we assigned different priorities to the eye, head, and body movement. The robot's eyes only degree of freedom, eye pan, is allowed to move at its maximum velocity. The movement of the head (nod and turn) is low-pass filtered in order to simulate moving the eyes quickly while the head does not follow every high frequency movement of the user. The body turning movement employs a low-pass filter with an even lower cut-off frequency and, hence, the robot always tracks the user with its eyes and head while only turning "lazily" during the waiting phase.

2) Missed Ball Detection and Acknowledgment: Unfortunately we cannot detect reliably if a ball is in the hand using visual or tactile feedback. If the ball is held in the hand it is partially occluded by the fingers and the detection by the vision system is unreliable. Alternatively, using the position sensors of the fingers, we could try to detect if the fingers are blocked by the ball when trying to close. As the fingers can only curl approximately 90° and the balls can deform significantly, this option also proved futile.

Rather than detecting catching successes, we look for failures. The robot stops briefly after the catch, before moving to the throwing position. At this time, we can detect whether the ball is below the catching plane, and is therefore a miss. Additionally, we can detect whether the ball is behind the robot, which also constitutes a miss. As a proof of concept, we implemented three simple animations when the ball drops below the catching plane: a shoulder shrug, shaking the head, and looking down. The animation is picked randomly when such a miss is detected. When the ball is detected behind the robot, we implemented an animation where the robot looks backwards towards the floor.

The vision-based missed ball detection works correctly for approximately 90% of the throws. We have roughly the same number of false positives and negatives. As the camera does not see a lot of area behind the robot and the robot can occlude the ball when it is between its arm and body, throws going behind the robot are sometimes not detected. Adding a ball contact sensor to the palm of the hand would allow more reliable detection.

H. Juggling

Juggling is achieved by speeding up the throwing/catching cycle. The state machine remains the same as in Fig. 5, however we remove the missed ball animations. Additionally, we trigger the start of the throwing state based on the timing of the incoming ball, leaving just enough time to release the ball in hand and catch a new one. The transition from catching a ball to moving to a throwing position originally left some time for the caught ball to settle in the hand. However, as speed is critical, we eliminated this delay and also shortened the original throwing pattern.

As tracking multiple balls simultaneously in the air confused the prediction algorithm, we developed a switching color mask. We used three different colored balls and fixed the pattern for throwing balls to the robot, for example green-red-yellow-green, etc. We then switched the vision system's color mask to the color of the next expected ball in the pattern. In this way, only the incoming ball is detected and the returning ball does not interfere with the prediction.

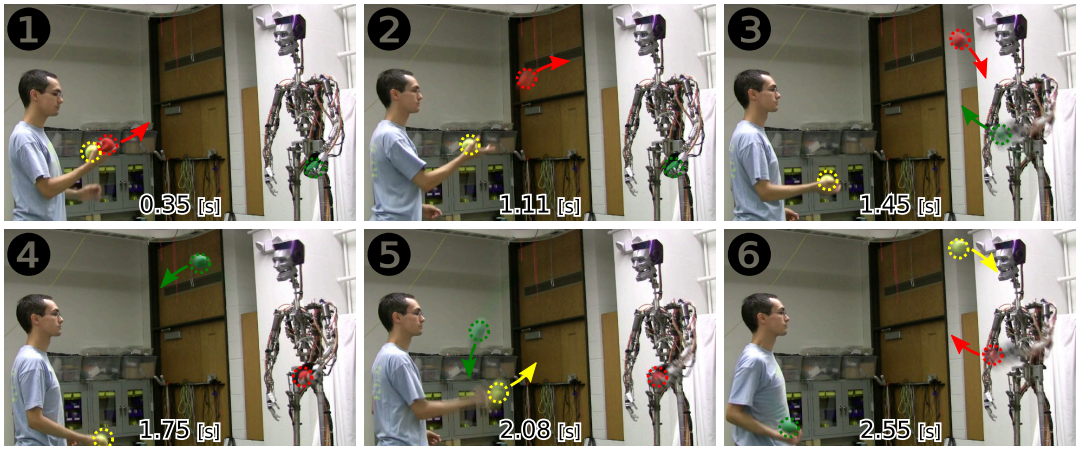

Fig. 9. Sequence showing three ball partner juggling. Balls are highlighted with colored circles. The robot starts with a green ball in hand, and tosses it up just before receiving the red. Red is thrown back as the yellow comes down, and so on.

III. Results

In order to quantify catching performance, we repeatedly tossed balls at the robot, recorded the anticipated catching position, and marked the success or failure of each catch. For each toss, the thrower stands on a 42cm stool, 2.5m away from the robot, and is asked to throw underhand to the robot's left hand. The stool was required to put the participants roughly at the same level as the robot, otherwise it was fairly hard to throw in the robot's catching region and the robot was "looking down" on the users. Fig. 7 visualizes the cumulative catching performance of 600 tosses. Four novice users (those with no prior experience with the robot), threw 200 of the tosses. Two experienced throwers, threw 400 balls attempting to cover the full catching region as much as possible. When the ball was thrown in the feasible catching region (red area of Fig. 6), catching success rate was 75% for both novice and experienced users. Overall success rate for all 600 throws was 47%, however many of these tosses were intended to sample the complete catching region, and not necessarily hit the robot's hand. Failures where usually caused by a combination of factors: low control frequency, low maximal velocities and accelerations, delays and low frame rate in the vision system, initially inaccurate predictions, and the linear model for the inverse kinematics.

To test the interactivity of our robot, we additionally invited six naive participants to "play catch" with our system, including an 11 year old child and 70+ year old elderly adult. Participants again stood on a stool, 2.5m away from the robot[2]. We handed them a ball and asked them to throw underhand to the robot's left hand. After roughly 5 to 15 initial tosses, all participants had figured out the catching region of the robot. All participants were also able to catch the incoming ball when it was thrown back, except for our elderly subject who did not even attempt catching as she was worried about her balance on the stool. All participants said the experience was fun and enjoyable. Some participants complained about standing on the stool, a limitation we hope to remove by lowering the robot into the floor or building a larger platform for participants. Some attempted the task without the stool, which was possible, but generally throwing was easier when on the same level as the robot.

As juggling requires greater skill, we brought in three skilled jugglers (members of Carnegie Mellon University's juggling club). This task proved to be more difficult as throws to the robot had to be fairly consistent and sufficiently high to allow the robot enough time to catch after a throw. We asked the jugglers to both throw and catch with only their right hand, while throwing to the robot's left hand. We also instructed the jugglers to throw the balls in a fixed color sequence (see Section II-H). After approximately 10-20 minutes of practice, all three jugglers were able to achieve at least three successful robot catches in a row with a three ball cascade pattern. One juggler managed to achieve four robot catches in a row. All three jugglers commented that the robot's limited catching region was the main difficulty in accomplishing the task, but felt they could improve their performance with additional practice.

An example sequence of throwing to the robot, catching and throwing back is shown in Fig. 8. Additionally, a cycle of juggling is shown in Fig. 9. Please see the video attachment for a complete visualization.

IV. Conclusions

We developed a platform for exploring human-robot physical interactions at a safe distance. As a test scenario, we use the context of a simple ball throwing and catching game. Our platform allows us to begin examining various storytelling scenarios within theme park or entertainment venues where guests can interact physically, but safely, with animatronic characters. Guests will be able to obtain a physical connection with characters and become participants in story events, creating a greater sense of immersion within fantasy environments.

Areas for improving our system include expanding the region where the robot is able to catch a ball. We used only

[2]The distance is limited by the robot's throwing capabilities as well as by the field of view and resolution of the camera.

arm and finger degrees-of-freedom for catching, but adding torso twisting and/or bending would expand the hand's range of motion. Adding more degrees of freedom and not restricting the region to a plane will require more complex models of the robot and proper motion planning. It would also be helpful to modify the catching hand such that the thumb or fingers can properly clasp and hold onto the ball, to minimize ball dropping after contact. A contact sensor or switch located on the palm would help to detect catch successes and failures, so we would not have to rely on the vision system for this purpose. The robot's throwing motion is currently predetermined and assumes the participant is two to three meters away. In the future, we hope to be able to adapt throwing to a variety of target locations and distances.

We found that subtle motion or animation cues significantly added to the participants' interaction with the robot. For example, simple head movements, to simulate ball and to look directly at the face of the participant, made the robot appear more natural and engaged with the task. Originally, we did not detect catching failures and it was quite jarring when the robot would throw back with an empty hand. We added four basic animations to acknowledge failed catches. It was a way to turn a failure into something unexpected and entertaining, as well as create an opportunity for new interaction. For example, after viewing an animation, two of our participants apologized to the robot for throwing so poorly, while two others scolded the robot. We are currently conducting an empirical study to evaluate the influence of such animations on the level of enjoyment of the interaction. We would like to further explore this type of interaction, perhaps by detecting if the catching failure was a result of a poor human throw or a robot error. Additional animations that convey encouragement, apology, goading, etc., could further enhance the engagement and entertainment experience. Perhaps the robot could talk or give instructions.

Juggling with our platform was an experiment that pushed both the physical and software limits of our system. Compared to existing high-performance robots, our robot is mechanically slow and has low bandwidth control. Delays from the vision system further compound the challenge. However, we observed interesting and unique human-robot interactions, where the human significantly adapted in response to the robot's limitations. When our skilled jugglers realized the robot would react more slowly than expected of a human partner, and could not complete throwing in time to catch a new ball, they would throw higher to give the robot more time.

We believe our platform affords further areas to examine new and unique real-world human-robot interaction, particularly in physical yet safe domains.

ACKNOWLEDGMENT

We thank Kareem Patterson and Ian MacVittie of Walt Disney Imagineering for setup and support of the A100 figure. We also thank Mouse Silverstein and Ross Stephens of Walt Disney Parks and Resorts for guidance on the software and control system interface. Sharon Hoosein, Moshe Mahler, and Rachel Maran assisted us on creating the art and video.

REFERENCES

[1] B. Hove and J.-J. E. Slotine, "Experiments in robotic catching," in *Proc. American Control Conf.*, 1991.

[2] M. Riley and C. G. Atkeson, "Robot catching: Towards engaging human-humanoid interaction," *Auton. Robots*, vol. 12, no. 1, pp. 119–128, 2002.

[3] R. Mori, K. Hashimoto, and F. Miyazaki, "Tracking and catching of 3D flying target based on GAG strategy," in *Proc. Int. Conf. on Robotics and Automation*, 2004.

[4] K. Deguchi, H. Sakurai, and S. Ushida, "A goal oriented just-in-time visual servoing for ball catching robot arm," in *Proc. Int. Conf. on Intelligent Robots and Systems*, 2008.

[5] T. Senoo, Y. Yamakawa, S. Mizusawa, A. Namiki, M. Ishikawa, and M. Shimojo, "Skillful manipulation based on high-speed sensory-motor fusion," in *Proc. Int. Conf. on Robotics and Automation (ICRA)*, 2009.

[6] G. Bätz, A. Yaqub, H. Wu, K. Kühnlenz, D. Wollherr, and M. Buss, "Dynamic manipulation: Nonprehensile ball catching," in *Proc. Mediterranean Conf. on Control and Automation*, 2010.

[7] B. Bäuml, T. Wimböck, and G. Hirzinger, "Kinematically optimal catching a flying ball with a hand-arm-system," in *Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[8] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters, "Trajectory planning for optimal robot catching in real-time," in *Proc. Int. Conf. on Robotics and Automation*, 2011.

[9] U. Frese, B. Bäuml, S. Haidacher, G. Schreiber, I. Schaefer, M. Hähnle, and G. Hirzinger, "Off-the-shelf vision for a robotic ball catcher," in *Proc. Int. Conf. on Intelligent Robots and Systems*, 2001.

[10] O. Birbach, U. Frese, and B. Bäuml, "Realtime perception for catching a flying ball with a mobile humanoid," in *Proc. Int. Conf. on Robotics and Automation*, 2011.

[11] O. Birbach and U. Frese, "Estimation and prediction of multiple flying balls using probability hypothesis density filtering," in *Proc. Int. Conf. on Intelligent Robots and Systems*, 2011.

[12] B. Bäuml, O. Birbach, T. Wimböck, U. Frese, A. Dietrich, and G. Hirzinger, "Catching flying balls with a mobile humanoid: System overview and design considerations," in *Proc. Int. Conf. on Humanoid Robots (HUMANOIDS)*, 2011.

[13] E. Aboaf, C. Atkeson, and D. Reinkensmeyer, "Task-level robot learning," in *Proc. Int. Conf. on Robotics and Automation (ICRA)*, 1988.

[14] E. Aboaf, S. Drucker, and C. Atkeson, "Task-level robot learning: juggling a tennis ball more accurately," in *Proc. Int. Conf. on Robotics and Automation (ICRA)*, 1989.

[15] A. A. Rizzi, L. L. Whitcomb, and D. E. Koditschek, "Distributed real-time control of a spatial robot juggler," *Computer*, vol. 25, no. 5, pp. 12–24, 1992.

[16] L. Yu and H. Ammar, "Analysis of real-time distributed systems: a case study," in *Proc. Midwest Symp. on Circuits and Systems*, 1992.

[17] S. Schaal and C. G. Atkeson, "Open loop stable control strategies for robot juggling," in *Proc. Int. Conf. on Robotics and Automation*, 1993.

[18] ——, "Robot juggling: An implementation of memory-based learning," *Control Systems Magazine*, vol. 14, no. 1, pp. 57–71, 1994.

[19] S. Schaal, D. Sternad, and C. G. Atkeson, "One-handed juggling: A dynamical approach to a rhythmic movement task," *Journal of Motor Behavior*, vol. 28, no. 2, pp. 165–183, 1996.

[20] T. Sakaguchi, Y. Masutani, and F. Miyazaki, "A study on juggling tasks," in *Proc. Int. Workshop on Intelligent Robots and Systems*, 1991.

[21] T. Sakaguchi, M. Fujita, H. Watanabe, and F. Miyazaki, "Motion planning and control for a robot performer," in *Proc. Int. Conf. on Robotics and Automation*, 1993.

[22] C. G. Atkeson. (2002) Humanoid robot juggling. [Online]. Available: http://www.sarcos.com/highperf_videos.html

[23] S. H. Yeo, M. Lesmana, D. R. Neog, and D. K. Pai, "Eyecatch: Simulating visuomotor coordination for object interception," in *ACM Transactions on Graphics (SIGGRAPH)*, 2012.