# Catching Flying Balls with a Mobile Humanoid: System Overview and Design Considerations

Berthold Bäuml, Oliver Birbach, Thomas Wimböck, Udo Frese, Alexander Dietrich, and Gerd Hirzinger

***Abstract*— A ball catching scenario with the mobile humanoid Rollin' Justin is presented. It can catch up to two simultaneously thrown balls with its hands, reaching a catch rate of over 80%. All DOF (degrees of freedom), i.e., the arms, the torso, and the mobile platform, are used for the reaching motion and the system works completely wirelessly using only onboard sensing. The task is demanding because of the necessary precision in space ($<$ 2cm) and time ($<$ 5ms) as well as its realtime character due to the short flying time ($<$ 1s). Fast perception, a good catching strategy and whole body path planning and control are important, but their tight interplay enabled by an appropriate system architecture is essential. The system overview presents the design considerations for extending Justin's versatility to this highly dynamic task. The key is not to radically change one component of the system but to do well-considered upgrades on all architectural levels, be it sensors, algorithms or middleware.**

## I. Introduction

Ball catching has a long history ([1], [2], [3], [4], [5]) as a challenging experimental task to help to test and further develop a number of robotic key technologies, like fast perception and estimation, sensor fusion, realtime path planning or realtime system architectures (including buses, computing resources, software middleware) to name only a few. To successfully catch a flying ball, a tight interplay of fast perception, a good catching strategy, body control and dexterity is needed to achieve the necessary precision in space and time.

### A. Related Work

Over the years the robotic systems used as well as the perception and planning methods got more and more complex.

In the famous and pioneering work [1], [2] the 4 DOF "WAM" arm, equipped with a gripper to grasp the ball, and an active vision system is used. The heuristic catch point selection chooses the closest point of the ball trajectory to the robot base and orients the gripper perpendicular to the trajectory. A Cartesian path is generated as a third order polynomial and executed by inverse kinematics running in the control loop.

The system presented in [4] has a 5 DOF arm on a stationary humanoid upper body, but only the arm is moving. It uses a "cooking basket" at the end effector for catching the ball and an active vision system. The inverse kinematics is solved by a neural network, which should lead to a human like movement behavior.

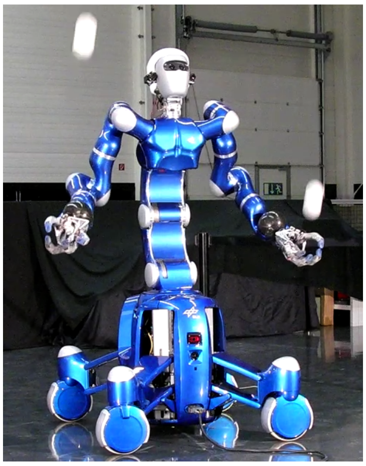

Fig. 1. Rollin' Justin [6] catching two simultaneously thrown balls. The mobile humanoid consists of an omnidirectional platform with variable footprint and an upper body with 19 active DOF, built from two DLR-LWR-III arms [7] with 7 DOF, a torso with 3 (+1 passive) DOF and a neck with 2 DOF, and is equipped with two 12 DOF DLR-Hand-II [8]. The system operates completely wirelessly using only onboard sensing (see Sec. V). Weight: upper body 45 kg; mobile base 150 kg.

In [5] the ball is perceived by a stationary stereo camera with 1m baseline mounted near the thrower. The ball is caught with a dexterous four-finger hand with 12 active DOF mounted on a 7 DOF lightweight robot. The planning of the optimal catch point and the generation of the joint paths is computed by solving a unified nonlinear optimization problem with nonlinear constraints. It generates a kinematically optimal catch motion, including even a simple geometry model of the robot and its workcell for collision avoidance.

### B. Contributions

In this paper we present ball catching with the mobile humanoid robot Rollin' Justin (Fig. 1) giving a substantially more in-depth description and discussion of the system compared to the video [9]. It can catch up to two simultaneously thrown balls with its hands with a success rate of over 80%. All DOF are used for the reaching motion, including the arms, torso and mobile platform (as 1 DOF translation). Further, the head joints are used to keep the ball in the cameras' field of view. The system operates completely wirelessly using only onboard sensing and an external compute cluster for path planning coupled by WLAN.

This exceeds previous work not only by the pure complexity of the used robotic hardware with its many DOF. In fact,

This work was partly supported by DFG grant FR2620/1-1.

DLR Institute of Robotics and Mechatronics, Münchnerstr. 20, 82234 Wessling, Germany berthold.baeuml@dlr.de

O. Birbach and U. Frese are with German Center for Artificial Intelligence (DFKI). 28359 Bremen, Germany

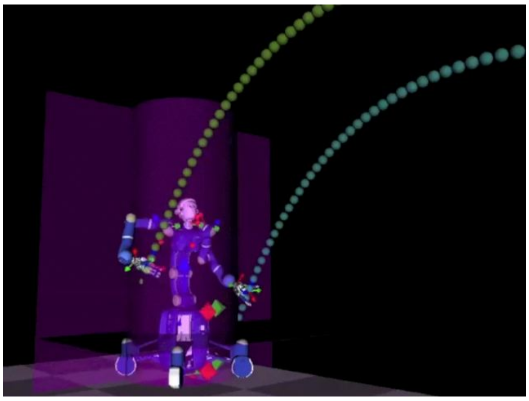

Fig. 2. The ball catching setup. The balls are thrown by a human from a distance of about 4-6m towards the robot with a speed of typically 7 m/s, resulting in a flight time of about 1s. The two ball trajectories ($\Delta t = 20\,\mathrm{ms}$) and Justin in the final catch configuration are shown. The (simple) collision model (virtual cylinder and walls), against which the planner tests both TCPs of the final catch configuration to avoid self-collisions, is depicted in pink. The ball has a diameter of 8.5 cm and a weight of 50 g resulting in significant air drag effects (for a 5 m throw $> 20\,\mathrm{cm}$ shorter flying distance as compared to a purely ballistic trajectory [3]).

the main challenge is that *everything is moving*: the camera system as well as the arms are mounted on a moving torso and mobile platform. So, one has to accurately keep track of all relative frames and timings, which is specially hard here because of, e.g., the elasticities of the robot’s lightweight structure or the not globally synchronized clocks of the many sensors, actuators and computers.

This paper serves two objectives. First, it provides a consistent overview of all components of the system. Second, the design decisions are described to extend the capabilities of Rollin’ Justin, which was originally built for bimanual (dexterous) manipulation, to also perform highly dynamic perception-planning-action-tasks like catching flying balls.

The maxim for this ”upgrade” of Justin was to change the system as little as possible, but to do well-considered improvements at all levels of the system architecture, be it sensors, perception, planning and control algorithms, computing and communication resources or the software middleware. We think the lessons learned here do not only apply to ball catching or humanoid robots. They hold more generally as a complex robotic system can only be made more versatile by such a holistic system architectural approach, because optimizing the hardware for one task (e.g., making the structure stiffer by adding material) usually sacrifices the performance for other tasks (e.g., low weight for mobility and safety).

## II. Setup and Challenges

### A. Setup

Fig. 2 shows the ball catching setup. The thrown balls are tracked by a head-mounted stereo camera system. Based on this, a (continuously improving) prediction of the balls’ trajectories is computed and sent to the planning module. Then the planner decides where, when, and in which configuration to catch the balls.

| | $q_{\min}[^\circ]$ | $q_{\max}[^\circ]$ | $\lvert\omega_{\max}\rvert[^\circ/\mathrm{s}]$ | $\lvert\tau_{\max}\rvert[\mathrm{Nm}]$ |
|---|---|---|---|---|
| arm 1 | -170 | 170 | 100 | 180 |
| 2 | -120 | 120 | 100 | 180 |
| 3 | -170 | 170 | 100 | 80 |
| 4 | -120 | 120 | 100 | 80 |
| 5 | -170 | 170 | 150 | 30 |
| 6 | -45 | 80 | 100 | 30 |
| 7 | -45 | 135 | 100 | 30 |
| head 1 | -45 | 45 | 330 | 30 |
| 2 | -20 | 50 | 205 | 50 |
| torso 1 | -140 | 200 | 110 | 180 |
| 2 | -90 | 90 | 110 | 229 |
| 3 | 0 | 150 | 110 | 229 |

TABLE I

Dynamical limits of Rollin’ Justin. The mobile base has $v_{\max} = 1.4\,\mathrm{m/s}$ and $a_{\max} = 5\,\mathrm{m/s^2}$.

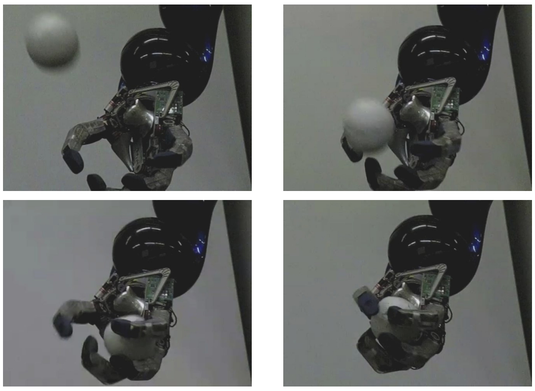

Fig. 3. Closeup of the hand catching a ball flying with a speed of about 5 m/s (upper left to lower right; first image immediately before the fingers start to move; $\Delta t = 26.5\,\mathrm{ms}$). Initially, the fingers are pre-shaped to form a little basket with a maximized opening cross section. After the ball has entered, the fingers are closed with maximum speed to finally cage the ball. The *catch frame*, that the planner uses as a virtual tool center point (TCP, one for each arm), lies in the middle of the opening cross section and its z-axis points anti-parallel to the ball’s flying direction. This leads to maximum robustness in compensating errors in space and time of the ball trajectory prediction as well as of the arm positioning. Finger parameters: length $L = 14\,\mathrm{cm}$, max. joint velocity $\omega_{\max} = 550^\circ/\mathrm{s}$.

### B. Challenges

*1) Low Latency:* Given the flight time of the balls of typically $< 1\mathrm{s}$ and the limited dynamical performance of Rollin’ Justin (cf. Tab. I), the robot has to start its reaching motion as soon as possible to get a reasonably sized *catch space*. Hence, the latency of the tracking, prediction and planning modules should be as low as possible.

*2) High Precision in Space and Time:* To successfully catch a ball, the hand positioning and finger-closing timing has to reach a precision of 2cm and 5ms, respectively. This is determined by the hand geometry, ball size and hand closing speed (see Fig. 3 for details), so that the ball does not hit the fingers of the open hand and does not bounce out of the hand again before the fingers can close.

*3) Moving Camera System:* To reach the necessary precision in space and time, the ball tracking module has to integrate all measurements of the ball’s positions during its flight until immediately before it is caught (because, the closer the ball comes to the robot, the more precise gets

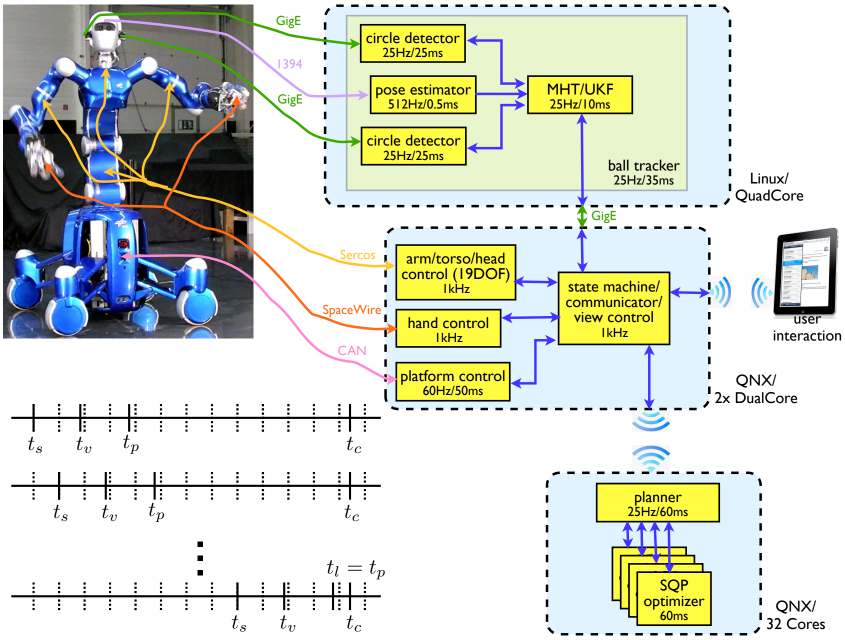

Fig. 4. System architecture. Rollin' Justin's sensors and actuators are coupled by a variety of bus systems to the onboard computing resource consisting of one Linux (Intel Core2Quad Q9000@2.00GHz) and two QNX (realtime OS) computers (Intel Core2Duo@2.4GHz), all connected by GigE network. An external cluster (32 CPU cores, 4x Dual-Quad-Core-Xeon) running QNX is coupled by WLAN. For all software modules, the rate at which they run, as well as their worst case processing time, is specified. When the cameras (running at 25Hz) take images at $t_s$, the data arrives $\Delta t_{v,c} = 40$ms later at the Linux computer, where the visual ball tracker module is triggered. After processing (worst case processing time $\Delta t_{v,p} = 35$ms), the resulting ball trajectory prediction is sent at $t_v = t_s + \Delta t_{v,c} + \Delta t_{v,p}$ to the coordinator module. The coordinator collects and distributes all data, controls the whole catching course (e.g., is the system active, catching or idle?), computes the desired head movements (see Sec. VI) and communicates with the planning module running on the external cluster. Based on the trajectory prediction, the planner computes the joint paths in $\Delta t_{p,p} = 60$ms and sends them back to the coordinator, where they arrive at $t_p = t_v + \Delta t_{p,p} + \Delta t_{p,c}$, with $\Delta t_{p,c} = 17$ms accounting for all delays of the back and forth transfers. The desired paths are then executed and the hand is closed at the planned catch time by means of the controller modules. The diagram in the lower left depicts the timing for two subsequent and the last camera frame ($\Delta t_{\mathrm{fr}} = 40$ms between dashed ticks).

the estimation of esp. the depth). This means, the robot is already performing its reaching motion including the torso and mobile platform and hence also the head. Therefore, the head motion relative to an inertial frame has to be tracked.

*4) Not Completely Cancelable Vibrations:* The upper body of Rollin' Justin is built following lightweight design principles with a number of advantages for a service robot. On the downside, the lightweight structure inevitably introduces elasticities and hence, vibrations, esp. for highly dynamical motions. These can not be completely canceled even by the applied elaborate control algorithms [10]. They are most dominant in the torso joints and, although they are measured, make it harder to estimate the movement of the head, and limit the precision with which the hand can be positioned as the path planner can not (easily) anticipate these vibrations.

*5) Not Fully Observable Kinematical State:* Beside the observable vibrations, there are non observable errors between the desired and the actually performed movement, esp. in case of the highly dynamic reaching motions during ball catching: a) the elasticities in the torso structure lead to "quasi-static" deflections, mainly depending on the final configuration of the arms; b) high accelerations and decelerations of the mobile platform can lead to wheel slippage, which corrupts the odometry; c) the legs of the mobile platform are equipped with dampers but without position sensors, and so, although the dampers are locked for ball catching, the precise (static) orientation of the whole humanoid is not known.

*6) Limited Computing Resources and Communication Bandwidth:* To be able to operate completely wirelessly only a WLAN-connection to the outside world is available and hence all computations requiring high bandwidth or low latency must be performed on the limited onboard hardware.

*7) No Globally Synchronized Clocks and Communication Latencies:* In complex robotic systems like Rollin' Justin, typically there are numerous sensors and actuators, which are connected by a variety of different bus architectures to the central computing resources. Usually the clocks of the I/O-devices, running at various rates, are not globally synchronized and the busses have different communication latencies. Moreover, the high computational demands can only be satisfied by a small cluster of network coupled onboard computers, again, each having its own clock.

All this makes it necessary but hard to establish system-wide consistent time stamps for the physical sensor and actuator events within the high precision needed. This is complicated by the many communication and processing steps involved until a camera measurement of the current ball position finally leads to a movement of the finger motors.

## III. System Architecture

The system architecture (see Fig. 4 for an overview) is the key for the tight interplay of all modules and for the low latencies and high timing precision (cf. Ch. 1,2,6,7).

*Parallel and Distributed Computing:* Even onboard, parallel and distributed computing resources are necessary: despite their high computational demands, the visual tracking and control modules have to run onboard because of the high data volume (100MB/s for both stereo images) and the high control rate (1kHz), respectively. By contrast, the planner is run on an external cluster, as only the ball trajectory predictions and resulting joint paths have to be transferred back and forth at the cameras' frame rate (25Hz).

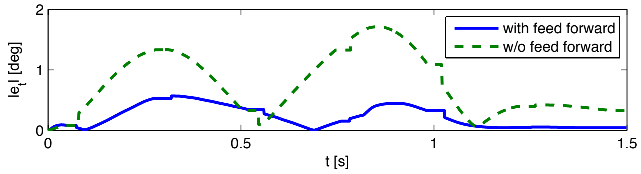

Fig. 5. The tracking accuracy of the torso joints can be significantly increased by the dynamic feed forward term $M_u(q_u)\ddot{q}_{u,d}$.

*Middleware:* All modules communicate by means of the aRD (agile Robot Development) [11] framework, which provides an easy to use, message-based middleware with low latency and realtime determinism (as far as the underlying OS does, e.g., $< 100\mu$s for transferring a 1.5KB packet between two QNX computers).

*Timestamps:* The system consists of many modules and the precision requirements with regard to timing precision are very high. Therefore, it is almost inevitable to use timestamps for all events throughout the system. Otherwise it would be very tedious and error-prone to keep track of all communication delays and processing times.

*Synchronized Computer Clocks:* For global timestamps on distributed computers, all clocks must be synchronized. The standard ntpd [12] synchronization is designed for wide area networks with often low bandwidth and high jitter. It is not appropriate here, because of its slow convergence after system startup in the order of 1h. This is mainly due to the slow update rate (maximally $\frac{1}{16}$Hz) and the used very robust, but slow converging jitter filtering heuristics. Therefore, a simpler timesync-demon has been developed, which takes advantage of the high bandwidth and low jitter of the local network (this holds even for the WLAN) and converges in a few minutes after system startup to a precision of $< 100\mu$s.

*De-jittering and Calibration of Delays:* Unfortunately, for some sensors, esp. the ones, which are connected to the non-realtime Linux computer, the time delay between actual measurement and arrival at the host is not deterministic, e.g., due to transmission delays generated by the FTDI USB to serial converter used by the XSens IMU. To nevertheless get precise timestamps of the sensors' measurements, de-jittering is required which is done by a Kalman filter with outlier rejection estimating the sensor's time offset and drift relative to the host's time. In addition, for the different sensors the fixed, but unknown latencies of the physical event and the arrival at the computer are calibrated (see Sec. V-B).

## IV. Control

### A. Overview and Challenges

In this section, the properties and control of the robot are highlighted. Rollin' Justin's mobile base is omnidirectional but non-holonomic. Before starting a consistent motion, the wheels have to orient in a way such that they align with the initial conditions of the motion trajectory. The mobile platform is kinematically controlled via a *dynamic feedback linearization* [13]. Thereby, it becomes possible to command the motion in the two translational directions, the rotation and the four tunable leg lengths separately. For ball catching the legs are locked via a special mechanism in the full extended configuration for maximum stability.

The light weight arm has a remarkable load to weight ratio of $1:1$ with a mass of 14 kg. Torque sensing is used on the one side for interaction with unstructured environments and the human, and on the other side to damp vibrations of the light weight structure [14]. The four fingered DLR-Hand-II has position sensors in all 12 DOF and, in addition, link-side torque sensors. All controllers run at 1kHz.

Catching a ball poses two main challenges for the control system. First, to accurately reach the desired catch point involving tracking of dynamic motions and vibration suppression. For a fast moving humanoid the dynamic effects due to the generated dynamic reaction forces, esp. at the torso and its base are much larger than, e.g., for a single arm mounted on a fixed base [10]. Furthermore, two torso joints are mechanically coupled to keep the robot chest upright. The coupling has considerable elasticity, adding vibrations to the system that have to be damped indirectly via the coupled joints. For good vibration suppression, the control gains on the position error cannot be very large. For those two reasons it becomes necessary to add feedforward tracking terms expressing the 17 DOF dynamics of the upper body. Second, the grasp control has to provide a fast closing motion that is compliant at the same time to reduce grasping and impact forces.

### B. Torso and Arms

For the upper body (subscript $u$) a joint space impedance controller is applied whose output is fed to a low-level torque controller that realizes the vibration damping [15]. The control law (subscript $d$ for desired) with the pos. def. PD matrices $K_{u,d}$ and $K_{u,p}$ is

$$\tau_{u,d} = M_u(q_u)\ddot{q}_{u,d} - K_{u,d}\dot{e}_u - K_{u,p}e_u + g_u(q_u), \qquad (1)$$

with $q_u = \begin{pmatrix} q_t^T & q_r^T & q_l^T \end{pmatrix}^T$ and $e_u = q_u - q_{u,d}$ the error in joint position, and $g_u(q_u)$ the gravity vector. Given the inertia matrices of the arms $M_{r,r}(q_r), M_{l,l}(q_l)$ and the torso $M_{t,t}(q_t)$, and given the attachment locations of the arms to the torso, the inertia matrix for the upper body can be derived as

$$M_u = \begin{bmatrix} M_{t,t} + M_{rr,t} + M_{ll,t} & M_{t,r} & M_{t,l} \\ M_{t,r}^T & M_{r,r} & 0 \\ M_{t,l}^T & 0 & M_{l,l} \end{bmatrix},$$

using a constraint formulation, where the terms $M_{xx,t}(q_u)$ are the contributions of the arm weight to the inertia of the torso and the terms $M_{t,x}(q_u)$ represent the dynamical coupling between torso and arms ($x = \{r, l\}$). Fig. 5 shows that the feed forward terms are effective, reducing the tracking error by more than a factor of 2.

### C. Hands

Grasping also uses joint impedance tracking control. The outer loop is a PD plus tracking controller of the form [16] (subscript $h$ for hands)

$$\tau_{h,d} = M_h(q_h)\ddot{q}_{h,d} - K_{h,d}\dot{e}_h - K_{h,p}e_h + g_h(q_h). \qquad (2)$$

The inertia matrices of all fingers are stacked in the blockdiagonal matrix $M_h(q_h)$. The position error is defined as $e_h = q_h - q_{h,d}$, with $q_h$ the joint positions and $q_{h,d}$ the desired equilibrium position. The PD matrices $K_{h,p}, K_{h,d}$ represent the stiffness and the damping behavior, respectively. The desired control torque $\tau_{h,d}$ is used as set point for a low-level torque controller.

Note, that even though the impedance parameters provide compliance, there is the time delay of 1 ms for the controller to react to the impact. The impact forces with a bandwidth larger than the one of the controlled system are seen directly by the structure and by the gears of the hand. In this case the light weight structure of the hand is another advantage because the bases of the fingers realize a mechanical base compliance that acts as a mechanical low pass filter.

## V. Visual Ball Tracking and Prediction

### A. Setup

The perception setup follows an anthropomorphic design. A pair of Prosilica GC1600 GigE cameras (synchronized, @1616 × 1220px, 1.5ms exposure time) is mounted on the sides of the head at a rather short baseline of 20cm. The high resolution improves precision, high framerate is unnecessary, since ball flight is well predictable. In combination with $f$ = 8mm lenses (Schneider Kreuznach CNG 1.4/8) each camera observes the scene in front of the robot at a field of view of $47^\circ$ horizontally and $36^\circ$ vertically. Lenses have been carefully chosen so that the angular extent of the scene is sufficient for a variety of ball trajectories and the prediction precision matches the requirement of 2cm.

The cameras are not static but move when the robot moves (s. Fig. 6b) and even shake from the reaction forces of moving the arms (s. Fig. 6a). As previously mentioned, these effects are reduced by the controller but not completely canceled and also not fully observable by the joint sensors (Ch. 3, 4, 5). Neglecting these vibrations would drastically reduce the precision. Consider, for example, tracking a ball while the head undergoes the motion of Fig. 6a. Conducted simulations (Fig. 6c) reveal that no compensation for this motion results in inaccurate predictions. Especially the rotation component perturbs the estimated velocity of the ball.

Our solution to this problem is to view the head-arm system as a self-contained catching device (cf. Sec. VI-B). It is somehow moved by the rest of the robot and we aim to obtain this motion solely from a head-mounted inertial measurement unit (XSens IMU @$512Hz$). Unfortunately, due to the dead-reckoning estimation approach of the pose, significant drift occurs even over short periods of time. To show how this affects the trajectory prediction accuracy we conducted another simulation (Fig. 6c). Here the motion is tracked by a simulated IMU assuming linear rotational drift and quadratic translational drift (overall 0.007m and $0.6^\circ$ after 0.6s, as observed in prior experiments [17]). Although the drift error almost accumulates to the magnitude of the shaking motion, the prediction accuracy is hardly affected complying with the requirement of 2cm. It is an interesting insight that the tracking and prediction problem is more forgiving to slow drift than to rapid disturbances. Two other advantages of using the IMU instead of forward kinematics are, that the attitude is observed and that the sensor setup is independent allowing separate development and evaluation.

### B. Calibration

Calibrating does not only involves the cameras, the IMU, and their geometric relation but also the relation to the kinematic chain. For the first, observations of a leveled checkerboard pattern at different head poses define the intrinsic parameters and the transformation between both cameras as well as the rotation between IMU and cameras. The translation is measured manually. For the second, visual markers attached to the robot's left and right hand are observed in different kinematic configurations of arm and head. This defines the transformation between the left (equivalently right) camera and the last head link of the kinematic chain. All calibration parameters are jointly estimated using MTKM [18], a framework for rapid specification of constraint graph problems (i.e. calibration) which are solved by least-squares.

Temporal delay between camera and IMU (Ch. 2, 7) is determined manually by matching predicted ball positions (from the tracking part) to the observed ball measurements in the images under rapid head motion. Similarly, relative delay to the kinematic chain (which itself is assumed to have no latency) is performed by matching head motion patterns.

### C. Tracking and Prediction

For tracking balls from camera images, we use a two-staged bottom up approach in which we first detect balls as circles and feed these measurements into a multiple hypothesis tracker (MHT) [19] handling multiple Unscented Kalman filters (UKF) as single target probabilistic models (cf. Fig. 7 for a view from the robot including results). Compared to simply fitting a parabola to triangulated 3D points, the UKF takes into account, that measurement uncertainties increase with distance and are larger in depth direction.

The motion model $\ddot{x}_b = g - \alpha|\dot{x}_b|\dot{x}_b$ accounts for gravity $g$ (measured by the IMU) and air-drag $\alpha$ (calibrated).

Circle detection is based on the average edge response along the circle. This system was first introduced in [20]. The detector had a bias for small circles which more often get high responses just by coincidence. By computing the likelihood that a detected response is caused by coincidence, this effect is compensated. Additionally, performance was optimized [17] for low latency on the limited on-board computing resources (Ch. 1, 6): Evaluation of circles has been parallelized using SIMD instructions and two cores process each image (OpenMP). Computation time and robustness of the tracker were improved by learning a prior on expected trajectories effectively preventing false-positive trajectories.

In numbers, circle detection of an image pair takes about 25ms. The ensuing tracking stage needs 5ms while idling and 10ms when tracking one or more balls. Prediction is performed by propagating the estimated ball position and velocity. The error over time is shown in Fig. 6d.

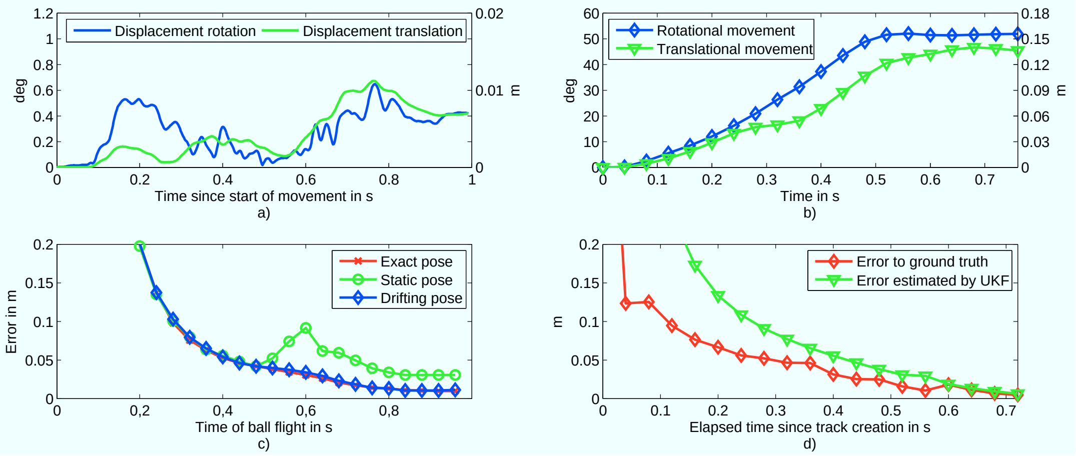

Fig. 6. **a)** Displacement of the robots head while catching a ball when the head is not actively moving. Shaking is solely induced from the reaction forces of a moving arm. **b)** Overall movement of the robot while moving head and torso. **c)** Error of predicted ball catch position over time relative to the robot's pose at catch time under different head pose estimation schemes (simulated, robot movement starts at about 0.4s). Exact pose allows accurate estimation of the predicted ball catch position over time (red). Neglecting the shaking head introduces large estimation errors (green). Surprisingly, using a low-cost dead-reckoning pose estimation which suffers from drift (e.g., MEMS-IMU), accuracy is only slightly imprecise in the beginning of the movement $(0.5 - 0.7s)$ but improves considerably as the ball approaches the robot (blue). In fact, the contribution of a measurement to the ball position relative to the robot in some moment is affected only by the IMU error accumulated since that measurement. Therefore, early measurements sustain larger errors than late ones but they are less precise anyway due to their distance. **d)** Prediction error over time and estimated accuracy obtained by propagating the covariance. Ground truth was obtained using an external tracking system.

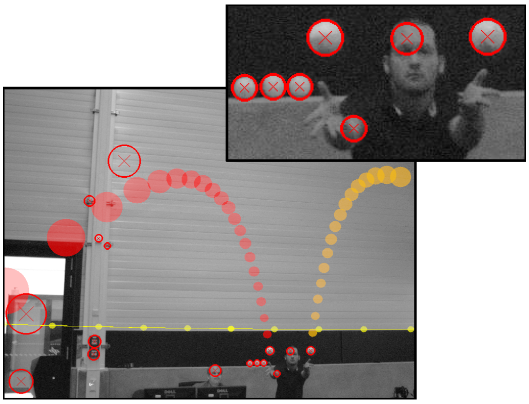

Fig. 7. View from one of the cameras mounted at the robot's head while a person is tossing two balls towards the robot. Results are marked in the image: camera orientation relative to $g$ is shown as an artificial horizon (yellow, with dots at every $5°$), circle detections are depicted as red circles while detected tossed balls are shown as their predicted trajectory through filled circles. Notice how not only the actual balls are detected but also other elements of the image which feature radial contrast.

## VI. Planning

### A. Optimization-Based Planner

Given the prediction of the balls' trajectories, the path planner has to decide where (catch point $x_c$), when (catch time $t_c$) and in which configuration (final joint angles $q_c$) to optimally catch the balls, while obeying constraints like, e.g., joint position limits, joint velocity limits or geometrical limits to avoid self-collisions (cf. Fig. 2). One DOF of the mobile platform is used for translation and to simplify notation, also named a 'joint'. Here an adapted version of the kinematically optimal path planner from [5] is used, where the planning problem is formulated as a nonlinear optimization with nonlinear constraints. The objective function is designed to make the motion "soft", i.e. having minimal joint accelerations, and to avoid local minima a parallel multi start technique is applied.

Other than the more general, sample-based path planners, like probabilistic roadmaps (PRM) and rapidly exploring random trees (RRT) [21], [22], which can cope with geometrically more complex scenes, the optimization-based planner directly computes smooth and optimal joint paths and does so very fast (planning time for a 7 DOF arm $\Delta t_p = 60$ms). The fast processing not only reduces the overall latency (Ch. 1), but also allows to re-compute the paths several times during the balls' flight, which is absolutely necessary as the trajectory predictions of the balls change significantly during the flight (cf. Fig. 6d).

### B. Kinematic Subchains

The planner uses a rigid body kinematics model of Rollin' Justin (cf. Fig. 9), ignoring all the significant effects as described in Ch. 4 and Ch. 5, like elasticities, vibrations or wheel slippage. A more precise model, however, would be way too complex for fast planning.

But this problem can be solved due to the kinematic structure of Rollin' Justin. On one side, the dominant sources of errors between the desired and actual motion is the kinematic subchain from the mobile platform to the torso's chest, whereas the two subchains from the hands to the head are very stiff and well modeled by a rigid body kinematics. On the other side, the cameras perpetually measure the balls' positions $x_b^H$ relative to the head frame $F_H$. So, the precision, with which the hand can be positioned relative to the ball at the catch time depends not on the full motion error of the platform→torso's chest chain, accumulated between the motion onset and the catch time $t_c$, but only on the much smaller error, accumulated during the much shorter time span

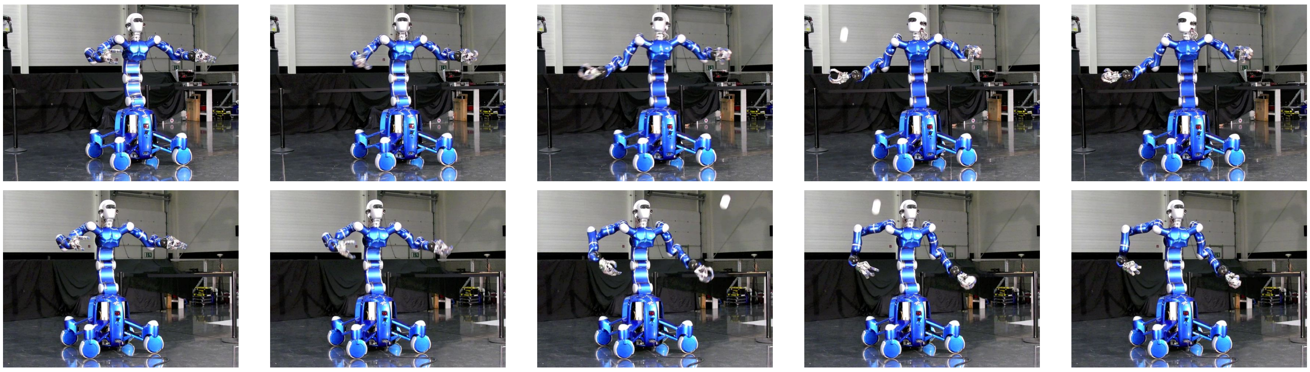

Fig. 8. Image sequences of Justin catching one (upper row) and two (lower row) balls (first image immediately before starting to move and $\Delta t = 200$ms). For the single ball case all DOF are moving, including the mobile platform, the torso and the head, whereas for the two ball case only the arms are used.

between the time $t_l^*$ of the last ball measurement before the actual catching and $t_c$! In the worst case, the difference is $t_c - t_l^* = \Delta t_{fr} + \Delta t_{v,c} + \Delta t_{v,p} + \Delta t_{p,c} + \Delta t_{p,p}$ (cf. Fig. 4). But the timespan can be further reduced, when taking advantage of the direct measurement of the head frame relative to the world by the IMU. Then only the processing time of the planner is relevant and for this $t_l$ time, $t_c - t_l = \Delta t_{fr} + \Delta t_{p,c} + \Delta t_{p,p}$.

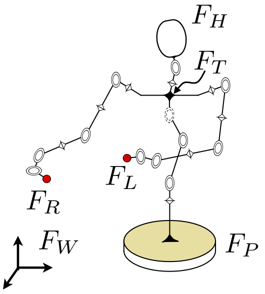

Fig. 9. Kinematics of Rollin' Justin with the frames: head $F_H$, torso's chest $F_T$, left $F_L$ and right $F_R$ hand, platform $F_P$ and an (arbitrary) world frame $F_W$.

From this it is clear that precise planning is only possible for the arms (i.e. torso's chest→hand chains), but it is even possible when the imprecise platform→torso's chest chain also moves. Therefore, for each arm an optimization-based planner is run for precise hand positioning, whereas the platform→torso's chest chain is used to extend the catch space by moving the 3 DOF of the torso and 1 DOF of the platform according to a simple heuristic, bringing the hand's start position closer to the predicted ball trajectory. In addition, the 2 DOF torso's chest→head chain is moved to keep the ball in the cameras' field of view during its flight by continuously pointing the head towards the ball positions according to the first valid prediction from the tracker module. For this a simple analytic inverse kinematics is applied and a limiter filter enforces the joint position and velocity limits.

The arm planner considers the torso's chest as its *static* base frame. But because in reality the chest is moving, the ball's trajectory $x_b$, as predicted from measurements at $t_s$ relative to the head frame, has to be transformed to the torso's chest frame at $t_c$:

$$x_b^{T(t_c)} = F^{T(t_c)}_{H(t_s)} x_b^{H(t_s)}, \tag{3}$$

where $F_A^B$ denotes the representation of frame $A$ in the coordinate system defined by frame $B$. As already discussed, to keep the influence of motion errors as little as possible, it is advantageous to use the most recent position *measurements* and only for times in the future the *desired* positions. It holds

$$F^{T(t_c)}_{H(t_s)} = \left(F^W_{T(t_c)}\right)^{-1} F^W_{T(t_p)} F^{T(t_p)}_{H(t_p)} \left(F^{H(0)}_{H(t_p)}\right)^{-1} F^{H(0)}_{H(t_s)},$$

where the fact is used that the world frame $W$ and the initial head frame (before any movement started) $H(0)$ are static. The head poses $F^{H(0)}_{H(t_p)}$ and $F^{H(0)}_{H(t_s)}$ are integrated by the IMU (see Sec. V). The other relative frames are obtained by forward kinematics from the joint positions $q = (q_T, q_H, q_L, q_R)$, where $q_T$ are the angles of the 3+1 DOF of the torso and platform, $q_H$ of the 2 DOF of the head and $q_L$, $q_R$ of the 7 DOF of the left and right arm ,respectively. An additional upper index specifies, if the angles are measured ($s$) or desired ($d$):

$$F^W_{T(t_c)} = F^W_{T(t_c)}(q^d_T(t_c)), \tag{4}$$

$$F^W_{T(t_p)} = F^W_{T(t_p)}(q^s_T(t_p)), \tag{5}$$

$$F^{T(t_p)}_{H(t_p)} = F^{T(t_p)}_{H(t_p)}(q^s_H(t_p)). \tag{6}$$

### C. Experimental Results

For the planner, the kinematic parameters from Tab. I and the timing parameters from Fig. 4 are used. Fig. 8 shows two catches, one with a single ball and one with two balls and in Fig. 10 the resulting joint paths for the single ball case are discussed (see also the video attachment).

## VII. Conclusion

Catching a thrown ball with a hand is a challenging task, esp. when done on a mobile humanoid, where "everything is moving" and only the limited onboard sensing and computing resources are available. Nevertheless, the presented humanoid Rollin' Justin finally reaches a catch rate of over 80% for balls thrown into the feasible robot's catch space. This is possible due to the performed upgrades on all architectural levels, e.g., adding an IMU, enhancing the control of the torso, intelligently taking the different kinematic subchains into account in planning and keeping track of all timings in the sub-millisecond range in software.

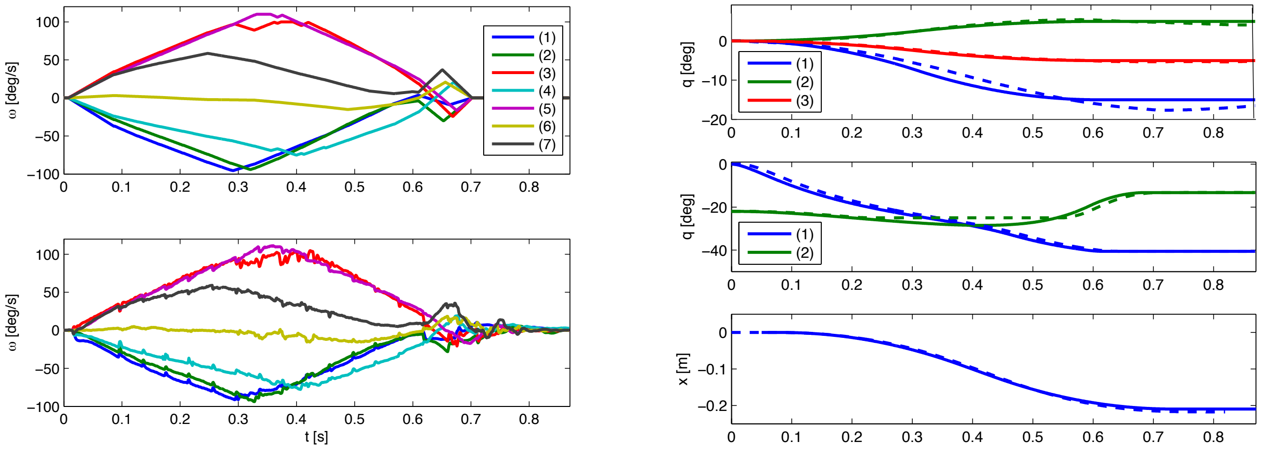

Fig. 10. Joint paths while catching a single ball (same as in upper row of Fig. 8), where all DOF participate in the motion. The robot starts to move at $t = 0$s, catches the ball at $t_c = 0.702$s. The last correction movement starts at $t_l = 0.680$s, which means, that the corresponding planning started at $t_v = 0.603$s. *Left Column*: Velocity profile $\dot{q}_R$ of the right arm. For the desired velocity (upper), the corrections due to the repeated re-planning based on the improving ball trajectory predictions are clearly visible as small kinks. Besides the noise, the measured profile (lower) very precisely resembles the desired one, proving, that the purely kinematic planner and its parametrization (cf. Tab. I) generates paths that do not exceed the dynamical capabilities of the robot. *Right Column*: Desired and measured (dashed) paths of the torso (upper), head (middle) and mobile platform (lower) joints. The deviations in the head motions are due to the limiter filter and not relevant (as long as the ball is still in the cameras' field of view), as only the measured angles are used in the transformations (cf. (6)). The motion errors of the torso and mobile platform, however, directly affect the final precision of the hand position, but only the accumulated error between $t_c$ and $t_v$ (the last time a new head measurement was integrated) counts: $\Delta q = \left(q^s_T(t_c) - q^d_T(t_c)\right) - \left(q^s_T(t_v) - q^d_T(t_v)\right)$, resulting in 1.5° for the torso's vertical joint (joint 1), which exhibits, as expected, the largest error, and 0.006m for the mobile platform.

Each of the improvements is important, as by only removing one, the catch rate significantly degrades.

In future work the catch space should be further increased by going closer to the robot's dynamical limits, but without damaging it. For this, instead of the kinematic, a dynamic planner has to be developed, which is challenging for such a high dimensional system under the hard timing constraints.

Acknowledgements: We thank the whole Justin hardware-team and Florian Schmidt for providing the timesync-demon.

## References

[1] B. Hove and J. Slotine, "Experiments in Robotic Catching," in *Proc. IEEE American Control Conference*, 1991, pp. 380–385.

[2] W. Hong and J. Slotine, "Experiments in Hand-Eye Coordination Using Active Vision," in *Proc. Fourth International Symposium on Experimental Robotics*, 1995.

[3] U. Frese, B. Bäuml, S. Haidacher, *et al.*, "Off-the-Shelf Vision for a Robotic Ball Catcher," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

[4] K. Nishiwaki, A. Konno, K. Nagashima, *et al.*, "The Humanoid Saika that Catches a Thrown Ball," in *Proc. lEEE International Workshop on Robot and Human Communication*, 1997, pp. 94–99.

[5] B. Bäuml, T. Wimböck, and G. Hirzinger, "Kinematically Optimal Catching a Flying Ball with a Hand-Arm-System," in *Proc. IEEE International Conference on Intelligent Robots and Systems*, 2010.

[6] C. Borst, T. Wimböck, F. Schmidt, *et al.*, "Rollin' Justin: Mobile Platform with Variable Base," in *Proc. IEEE International Conference on Robotics and Automation*, 2009, pp. 1597–1598.

[7] G. Hirzinger, N. Sporer, A. Albu-Schäffer, *et al.*, "DLR's Torque-Controlled Light Weight Robot III - are we Reaching the Technological Limits now?" in *Proc. IEEE International Conference on Robotics and Automation*, 2002, pp. 1710–1716.

[8] J. Butterfaß, M. Grebenstein, H. Liu, *et al.*, "DLR-Hand II: Next Generation of a Dextrous Robot Hand," in *Proc. IEEE/RSJ International Conference Robotics and Automation*, 2001, pp. 109–114.

[9] B. Bäuml, F. Schmidt, T. Wimböck, *et al.*, "Catching Flying Balls and Preparing Coffee: Humanoid Rollin'Justin Performs Dynamic and Sensitive tasks," in *Proc. IEEE International Conference on Robotics and Automation*, 2011.

[10] T. Wimböck, D. Nenchev, A. Albu-Schäffer, *et al.*, "Experimental Study on Dynamic Reactionless Motions with DLR's Humanoid Robot Justin," in *Proc. IEEE International Conference on Intelligent Robots and Systems*, 2009.

[11] B. Bäuml and G. Hirzinger, "When hard realtime matters: Software for complex mechatronic systems," *Robotics and Autonomous Systems*, vol. 56, no. 1, pp. 5–13, 2008.

[12] [Online]. Available: http://www.ntp.org/

[13] P. Giordano, M. Fuchs, A. Albu-Schäffer, *et al.*, "On the Kinematic Modeling and Control of a Mobile Platform Equipped with Steering Wheels and Movable Legs," in *Proc. IEEE International Conference on Robotics and Automation*, 2009, pp. 4080–4087.

[14] A. Albu-Schäffer, O. Eiberger, M. Fuchs, *et al.*, "Anthropomorphic Soft Robotics - from Torque Control to Variable Intrinsic Compliance," in *International Symposium on Robotics Research*, 2009.

[15] Ch. Ott, A. Albu-Schäffer, A. Kugi, *et al.*, "On the Passivity Based Impedance Control of Flexible Joint Robots," *IEEE Trans. Robot.*, vol. 24, no. 2, pp. 416–429, April 2008.

[16] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[17] O. Birbach, U. Frese, and B. Bäuml, "Realtime Perception for Catching a Flying Ball with a Mobile Humanoid," in *IEEE Proc. International Conference on Robotics and Automation*, 2011.

[18] R. Wagner, O. Birbach, and U. Frese, "Rapid Development of Manifold-Based Graph Optimization Systems for Multi-Sensor Calibration and SLAM," in *Proc. IEEE International Conference on Intelligent Robots and Systems*, 2011.

[19] I. J. Cox and S. L. Hingorani, "An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and Its Evaluation for the Purpose of Visual Tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 2, pp. 138–150, 1996.

[20] O. Birbach and U. Frese, "A Multiple Hypothesis Approach for a Ball Tracking System," in *Computer Vision Systems*, ser. Lecture Notes in Computer Science, J. Piater, B. Schiele, and M. Fritz, Eds., vol. 5815. Springer, 2009, pp. 435–444.

[21] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[22] H. Gonzalez-Banos, D. Hsu, and J. Latombe, "Motion planning: Recent developments," in *Automous Mobile Robots: Sensing, Control, Decision-Making and Applications. CRC*, S. u. F. L. GE, Ed., 2006.