Kinematically Optimal Catching a Flying Ball with a Hand-Arm-System

Berthold Bäuml, Thomas Wimböck and Gerd Hirzinger

***Abstract*— A robotic ball-catching system built from a multi-purpose 7-DOF lightweight arm (DLR-LWR-III) and a 12 DOF four-fingered hand (DLR-Hand-II) is presented. Other than in previous work a mechatronically complex dexterous hand is used for grasping the ball and the decision of where, when and how to catch the ball, while obeying joint, speed and work cell limits, is formulated as an unified nonlinear optimization problem with nonlinear constraints. Three different objective functions are implemented, leading to significantly different robot movements. The high computational demands of an online realtime optimization are met by parallel computation on distributed computing resources (a cluster with 32 CPU cores). The system achieves a catch rate of $> 80\%$ and is regularly shown as a live demo at our institute.**

I. Introduction

Catching a thrown ball with a hand is not easy – neither for humans nor for robots . It demands for a tight interplay of skills in mechanics, control, planning and visual sensing to reach the necessary precision in space and time. Because of this, ball catching has been used for almost 20 years now as a challenging benchmark system to develop and test robotics key technologies [1], [2], [3], [4], [5], [6]. In all the works the general setup is in principle the same: a stereo vision system tracks the ball and predicts the balls trajectory, then the point and time, where and in which orientation the robot should intercept the ball on its trajectory, is determined. Next, the robot configuration to reach the catch point is computed and finally a path is generated, which brings the robot from its start configuration to the desired catch configuration.

A. Related Work

In the famous and pioneering work [1], [2] the 4 DOF "WAM" arm 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 an inverse kinematics running in the control loop.

A system with a 5 DOF arm on a humanoid robot (only the arm is moving) with a "cooking basket" at the end effector for catching the ball and an active vision system is presented in [3]. The inverse kinematics is solved by a neural network, which should lead to a human like movement behavior.

In our previous work [4] we used the 7 DOF DLR-LWR-II arm with a small basket at the endeffector and a stationary stereo camera and image processing system built from cheap "off-the-shelf" components (PAL cameras with 50Hz and standard PC). The catch point is determined with a simple heuristic that balances two goals: a) choose a point near the robot, so that the robot has not to move very far and can, although it is not very fast, reach the catch point in time and b) choose a catch point far away from the robot, to prohibit folding up of the robot and, hence, running into the joint limits. Next, an inverse kinematics computes the catch configuration, taking into account, that the orientation of the catching basket opening should be perpendicular to the ball trajectory. Finally, an interpolator generates the joint paths with a trapezoidal velocity profile (in joint space).

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

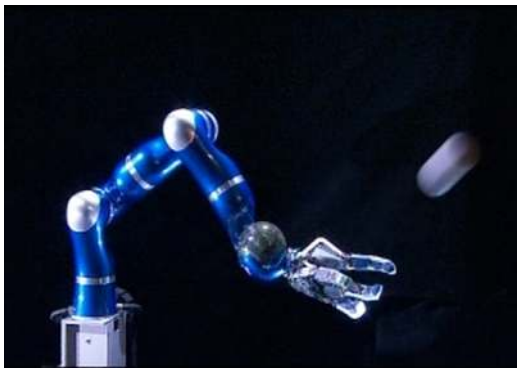
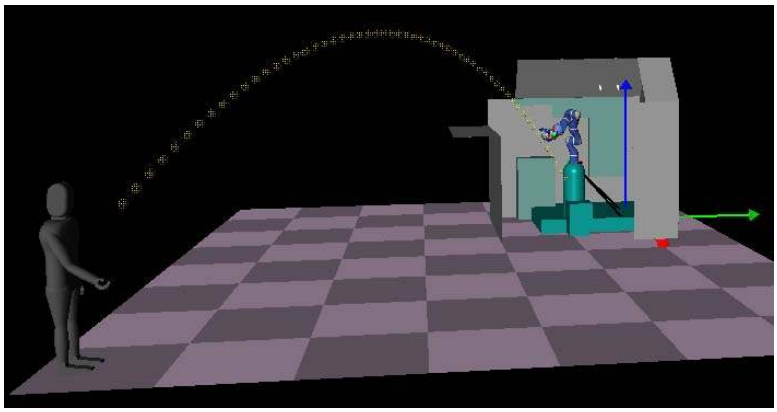


Fig. 1. The hand-arm system catching a ball. The system is built from the DLR-LWR-III arm with $N = 7$ DOF [7] and the 12 DOF DLR-Hand-II [8]. All joints are equipped with torque sensors and are impedance controlled. Weight: LWR-III 14 kg; Hand-II 2 kg.

In [5] the focus is on investigating motion primitives for human like path generation. The catching is done with an arm of a humanoid robot (only the arm moves) equipped with a base ball glove to compensate for only determining the catch point and not the endeffector orientation. The catch point is chosen to be the intersection of the ball trajectory with a horizontal plane at a given height and an inverse kinematics computes the catch configuration.

A 6DOF robot arm, especially built for ball catching, with a basket is described in [6]. The arm is built from commercial off-the-shelf parts with exceptionally high dynamics with joint velocities up to 470 deg/s and power consumption of up to 5kW, however. The main focus of the work is on tele-operated ball catching, but an autonomous catching using a stereo vision system was also implemented as a benchmark. A heuristic chooses the catch point as the intersection of the ball trajectory with a plane in front of the robot and an analytical inverse kinematics computes the catch configuration of the robot.



2. Overall geometrical setup (each tile of the floor is 1m x 1m). The ball is thrown by a human from a distance of about 5 m onto the robot with a speed of typically 7 m/s, resulting in a flight time of about 1s. The robot (depicted in standard starting configuration) is placed inside a working cell (mockup of a spacelab module) with walls, that lie *inside* the kinematic work space of the robot. The stereo camera system (two PAL cameras with 50Hz rate for video fields) is placed above the throwing person at 2 m height and has a vertical base length of 1 m. The ball has a diameter of 8.5 cm and a weight of 70 g resulting in significant air drag effects (for a 5 m throw $>$ 20 cm shorter flying distance as compared to a purely ballistic trajectory; see [4] for details)

B. Contributions

In this paper we present a ball catching setup, which significantly differs in two aspects from previous work. First, we use a dexterous multipurpose four-fingered hand (DLR-Hand-II, see Fig. 1 and [8]) to grasp the ball. This is challenging for the hardware and controllers as the hand has to be able to close fast enough and in addition it has to be robust enough, so that the impact of the ball does not damage the highly complex mechatronic device. Moreover, this is also challenging for the planning algorithms for the arm movements, because not only the position but also the orientation of the hand relative to the ball trajectory has to be very precisely determined, so that the ball can be grasped and does not hit (and damage) the fingers in a disadvantageous configuration.

The second new aspect is, that, instead of using separate steps for catch point selection, catch configuration computation and path generation, we present an *unified approach*, which subsumes all three steps in a single, nonlinear optimization problem with nonlinear constraints. This not only gives rise to a better overall performance (e.g. bigger reachable workspace and faster thrown balls can be caught) as one can get closer to the dynamical limits (maximal velocity and acceleration) of the robot, but also allows to optimize for different criteria, which leads to significantly different catch behaviors.

II. Overall Setup and System Architecture

Fig. 2 shows the geometrical setup, which is almost the same as in our previous work [4]. Also the same visual tracking and Extended Kalman Filter based prediction of the ball trajectory are used providing a new prediction every 20ms. The visual tracking is quite robust with successfully tracking $>$ 80% of the thrown balls, but the prediction varies (see Fig. 3) over time and typically reaches only about 100ms before the catch the necessary precision of $<$ 2cm and $<$ 5ms (for details on the grasp precision demands see Sec. III).

This makes it necessary, that the planning algorithm replans the robot movement every 20ms with as little delay as possible to keep up with the changing trajectory prediction. Because our planning algorithm is based on a nonlinear optimization with nonlinear constraints (see Sec. IV), it is

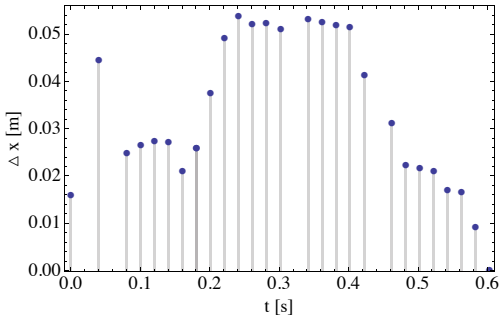
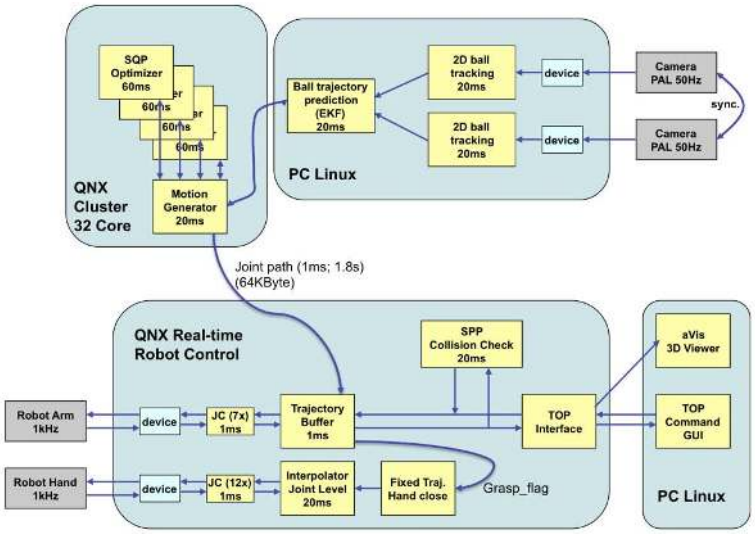


Fig. 3. Error of the ball trajectory prediction during a throw. The distance between the final catch point x_c and the catch point prediction $x_{b,t}(t_c)$ made from the measurements until time t is plotted. $x_{b,t}$ is the ball trajectory prediction as it is known at time t and t_c is the final catch time.

computationally demanding, even though we use the highly efficient SQP (sequential quadratic programming) method [10] in an implementation taken from [11]. Running on up-to-date hardware – a single core of a 2.2Ghz Quad-Core-Xeon – the worst case runtime is 60ms with 10ms on average. Hence, at least three CPU cores have to be used in parallel in a round robin scheme, where each new ball trajectory prediction is assigned to a currently idle core, to not omit any prediction.

Finding the first optimal movement path given the first ball trajectory prediction is computationally even more demanding, as no previous solution is present, which could be used for initializing the optimizer and which would have to be only slightly adapted to account for the slight changes in the prediction. Because of the many local minima, which arise mainly due to the nonlinear constraints (see Sec. IV for details), the optimization has to be executed with many different starting solutions, to find the optimal (or at least a good) solution with high probability. Therefore, a compute cluster with up to 32 CPU cores is used to execute a parallel multi start search for an optimal first solution. The search is stopped after 60ms and results in about 100 optimized solutions on average, including with high probability a nearly globally optimal movement path as has been approved by batch runs with more than 10000 start solutions.



4. System architecture showing the processing modules and their mapping to the computation hardware as well as the cycle times. The flying ball is observed by a stereo camera system (standard PAL cameras, 50Hz video field rate) which sends every 20ms the two video fields to a Linux PC (Pentium IV, 2GHz), where the 2D-Tracking modules and the estimation (with Extended Kalman Filter, EKF) and prediction modules for the 3D ball trajectory are running (overall delay from camera shutter until prediction finished is 30ms). The resulting prediction is sent to the online realtime planning module running on a cluster of realtime computers (32 CPU cores built from 4x Dual-Quad-Core-Xeon servers running the QNX [9] realtime OS). The planner consists of two parts: the "compute slaves" (one for each CPU core), which are actually executing the sequential quadratic programming (SQP) optimization algorithm in parallel, and the "move generator" which coordinates the slaves and communicates with the vision and the robot control system. After delay of 60ms the planner outputs a path with 1ms time resolution and 1.8s duration (the maximal allowed duration of a catch movement), which contains for each time step the desired arms joint angles and a flag for the hands grasp state (i.e. open or close). The robot arm and hand controllers, running at 1kHz on a QNX PC (Core2Duo, 3GHz), then execute the path and close the hand at the desired catch time. The overall delay from camera shutter to robot movement is 90ms.

The system architecture (see Fig. 4) has to provide distributed computational resources while meeting hard realtime constraints for the parallel planning as well as the robot controllers. For the communication between the software components running on the diverse computers we use the aRD (agile robot development) software concept [12], which we especially developed for applications with complex mechatronic components with high computational demands under hard realtime constraints. By using Gigabit Ethernet connections a worst case delay of $<$ 1ms even for transmitting the largest data packets in the system (the arm movement path for the 7 joint angles) with a size of 64KByte could be achieved.

III. HAND

The robot system is composed of the DLR-LWR-III and the DLR-Hand II. In this section the properties and control of the robotic subsystems are highlighted. The light weight arm has a remarkable load to weight ratio of $1:1$ with a mass of 14 kg. The torque sensing is used on the one side for interaction with unstructured environments and the human, and on the other side it is used to damp vibrations of the light weight structure [13]. The hand has four fingers with 3 DOF each, resulting in overall 12 DOF (c.f. Fig. 1). In addition to the position sensors the link-side torques are measured as well. The control law runs on a Core2Duo, 3GHz with a controller sample time of 1 ms.

The catching of a ball poses basically two main requirements to the control system. First, the accurate reaching of the desired catch point including tracking capabilities for dynamic motions and vibration suppression. Second, the grasp control has to provide a fast closing motion that is compliant at the same time to reduce grasping and impact forces.

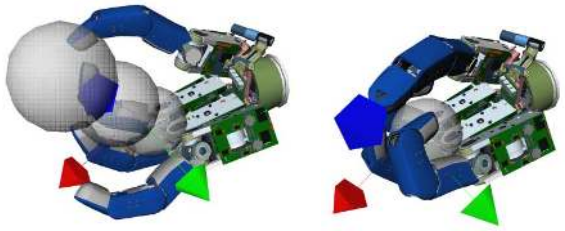


Fig. 5. The hand configurations before (left) and after (right) catching the ball and the *catch frame* that acts as tool center point (TCP) of the robot. In the catch frame the z-axis (blue) represents the flight direction of the ball. It is chosen such that the ball will approach the hand from its open side, i.e. that the ball passes the opening between the widely open ring finger and the thumb. Finally, hitting the proximal phalanxes of the index and the middle finger. Furthermore, note that the thumb and the index finger are already configured such that the ball can barely escape between them. In the closed configuration the hand realizes a spherical grasp. In order to cage the ball the thumb is moved into the direction of the ring finger. In this way the flight channel to enter the hand is closed. Note that the CAD drawing is missing a palm element mounted on the base of the palm that further restricts the motion of the ball (compare the real hand in Fig. 8 with real grasping).

For the arm a joint space impedance control is applied whose output is fed to a low-level torque controller that realizes the vibration damping [14]. For the grasping also a joint impedance tracking controller is applied. For the outer loop of the control law a PD plus tracking controller of the form

$$\tau_d = M_{q_h}\ddot{q}_{h,d} - K_d\dot{e} - K_p e + g(q_h, q) \qquad (1)$$

is applied [15]. The vector $\tau_d \in \mathbb{R}^{12}$ contains the desired joint torques. The inertia matrix of all finger joints is summarized in stacked notation in the blockdiagonal matrix M_{q_h}. The position error is defined as $e = q_h - q_{h,d}$, with q_h the joint positions of the hand and $q_{h,d}$ the desired equilibrium position. The PD matrices K_p, K_d represent the stiffness and

the damping behavior respectively. The term gravity term $g(q_h, q)$ is compensated while the Coriolis and centrifugal terms are neglected, with $q \in \mathbb{R}^7$ the joint angles of the arm. The desired control torque τ_d is used as set point for a low-level torque controller which is based on the joint torque measurement, and offline estimated static and viscous motor friction parameters. The joint velocity is estimated by means of an observer that is described in [16].

Note, that even though the impedance parameters provide compliance there is the time delay of 1ms 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 mechanical low pass filter[1].

Besides the hand control it is a difficult problem how to select the grasping strategy. That includes the choice of impact point in the hand, which is closely related to the selection of the so called *catch frame* that acts as tool center point (TCP) of the robot (compare Fig. 5). Depending on this choice of catch frame one needs to find a preshape of the hand that is furthermore a function of the estimation accuracy of the ball trajectory w.r.t. the arm coordinate system (here: 2 cm). The preshape is selected such that the ball including its position accuracy can enter the hand. For the closing we apply a caging strategy since the transitions within the hand has multiple contacts with the fingers and the palm. It is critical that the hand starts closing with a timing resolution of less than 5ms[2] in order to avoid that the ball hits the fingers or that the ball bounces back out of the hand. At this point the preshape and the final grasp configuration are optimized manually by using simulations in virtual reality and by the experiences during the experiments. Currently, we are working on methods to derive the methods to obtain such strategies automatically.

IV. Kinematically Optimal Realtime Planner

When a ball is thrown at $t_0 = 0$, the planner has to solve the following problem: given the start configuration of the robot$\boldsymbol{q}(0) = \boldsymbol{q}_0$ and start joint velocities $\boldsymbol{\omega}(0) = \boldsymbol{\omega}_0$, at which time t_c and in which configuration $\boldsymbol{q}(t_c) = \boldsymbol{q}_c$ with $\boldsymbol{\omega}(t_c) = 0$ should the robot intercept the ball on its trajectory, while obeying constraints like e.g. joint limits, joint velocity limits or geometrical workcell limits ? Note, that the robot does not have to stand still at be beginning, which is especially important to allow for recommanding, when the ball trajectory prediction changes over time.

The state of the art in general path planning are sample based methods, like probabilistic roadmaps (PRM) and rapidly exploring random trees (RRT), as presented e.g. in [17], [18] and [19]. As is noted in [17], these methods are especially useful to find a first feasible solution in problems, where the working cell is cluttered with geometrical obstacles. If a path is needed, which in addition is optimal with regard to e.g. time, energy or path length, the previously found solution is used in a second step as a starting point in a general nonlinear optimization like in [20]. For our ball catching setup, however, the first step of finding a path by means of sample based methods is on the one hand not necessary, because of the quite simple geometrical constraints mainly at the bounds of the working cell, and on the other hand not feasible, as those algorithms are computationally too demanding to be solved under the tight timing constraints we have to meet. Therefore, we formulate the problem directly as a nonlinear optimization with nonlinear constraints and avoid the problem of local minima by using a parallel multi start technique.

[1]Certainly, there exist configurations in which large forces can be exerted on the finger. This is the case when the ball hits the finger in the direction from the fingertip to the finger base.

[2]Assuming that the ball flies within the hand about 0.03 m with a velocity of 6 m/s the time duration of 5 ms is obtained.

	$q_{\min}[^\circ]$	$q_{\max}[^\circ]$	$\lvert\omega_{\max}\rvert[^\circ/\mathrm{s}]$	$\lvert\tau_{\max}\rvert[Nm]$
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

TABLE I

Limits of the LWR-III arm as used in the planner. The maximal acceleration is set to $|a_{\max}| = 860\circ/\mathrm{s}^2$ for all joints.

A. Assumptions

Because solving even "only" the general nonlinear optimization problem accurately is still computationally too demanding to be executed in realtime, some assumptions and approximations utilizing details of our setup have to be made to simplify the problem significantly while still getting a nearly optimal solution for the original problem.

1) Kinematically Planning: Although to catch a ball the robot has to move fast and in general its dynamics plays an important role for generating an optimal movement path, we chose to do a purely kinematically optimization, i.e., the objective function and constraints depend only on $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ and not on the joint torques $\boldsymbol{\tau}$.

This approximation is not too bad for the LWR-III, because for fast motions the joint velocity limits dominate the robot behavior: e.g. if the robot has to move in 1s (the balls typical flight time in our setup) as far as possible, only 0.1s are needed for accelerating and decelerating respectively, but 80% of the time the robot moves at maximum joint velocity. That means, that the details, of how the robot accelerates/decelerates are not very important for the overall performance, as long as the maximally allowed accelerations are conservatively chosen, to never exceed the torque limits of the robot (see Tab. I for the chosen parameters and limits).

2) Trapezoidal Joint Velocity Ramps: We restrict the robot movements to trapezoidal velocity ramps, because, as argued before, the precise characteristics of the acceleration and deceleration phases are not essential for the overall

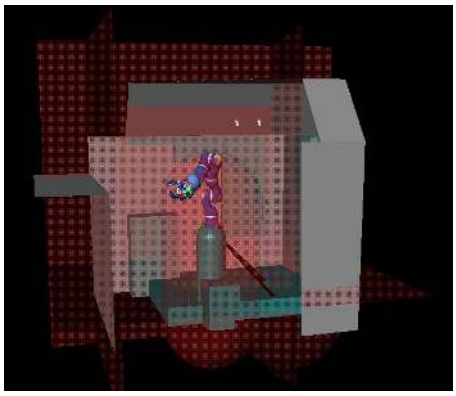


Fig. 6. Collision model. The TCP of the robot $\boldsymbol{x}_\mathrm{R}$ (the origin of the small coordinate system near the hand; see also Fig. 5) is tested against this model by calculating the smallest distance between the TCP position and the collision objects. The model consists of five planes (floor, ceiling and three walls) and three cylinders with spherical caps (robot platform and the two stabilizing rods in the back). The platform cylinder is "blown up" to additionally avoid self collisions of the robot. When calculating the distance to the planes an additional safety distance of 0.3m is subtracted to account for stretched out fingers.

performance and trapezoidal ramps can be easily analytically handled. Moreover, for the objective functions, we present here (see Sec. IV-B.1), the trapezoidal ramps are even optimal.

Using the step function

$$\theta(t) = \begin{cases} 1 & \text{if } t \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

with $\int_0^t dt' \theta(t') = t\theta(t)$ and the definition $\hat{\theta}(t,t_1,t_2) = \theta(t_1 - t) - \theta(t - t_2)$, with $t_1 \leq t_2$, the trapezoidal ramp for a single joint can be written as:

$$\begin{aligned} \ddot{q}(t) &= a\hat{\theta}(t,t_1,t_2), \\ \dot{q}(t) &= \omega_0 + at\,\hat{\theta}(t,t_1,t_2), \\ q(t) &= q_0 + \omega_0 t + \frac{a}{2}t^2\hat{\theta}(t,t_1,t_2). \end{aligned} \tag{2}$$

The acceleration phase with $a_\mathrm{acc} = a$ ends at t_1, then a phase with $\dot{q} = \omega_\mathrm{max}$ lasts until t_2, when the deceleration phase with $a_\mathrm{dec} = -a$ starts. There are two cases: first, $t_1 = t_2$, i.e. a triangle ramp, and second, $t_1 < t_2$, i.e. a "full" trapezoidal ramp, where, because of $\dot{q}(t_1) = \omega_\mathrm{max}$, it holds $t_1 = \frac{\omega_\mathrm{max} - \omega_0}{a}$. This means that the ramp in both cases is completely determined by only the two parameters a and t_2.

3) Collision Avoidance Heuristics: Only the position of the TCP of the robot (see Sec. 5) is tested against a collision model of the environment and even that is done only for the final catch configuration $\boldsymbol{q}_\mathrm{c}$. That this guarantees a collision free path of the whole robot during the whole movement, is established by making the objects in the collision model bigger than the physical objects (see Fig. 6), and by the limited task that has to be fulfilled, i.e. to catch a ball coming in from the front.

B. Optimization Problem

With this assumptions the nonlinear optimization problem with nonlinear constraints can now be written as:

$$(\boldsymbol{q}_\mathrm{c}, t_\mathrm{c}) = \min_{(\boldsymbol{q},t)\in\mathscr{S}} H(\boldsymbol{q},t), \tag{3}$$

$$\begin{aligned} \mathscr{S} = \{(\boldsymbol{q},t) \in \mathscr{R}^N \times \mathscr{R} : \quad & h_i(\boldsymbol{q},t) = 0, i = 1 \ldots N_h, \\ & g_j(\boldsymbol{q},t) \geq 0, j = 1 \ldots N_g\}, \end{aligned}$$

with an objective function H, equality constraints h_i and inequality constraints g_j. Note, that the optimization space is only $N+1$ dimensional, because due to the assumptions (especially the trapezoidal ramps assumption), the robot movement is completely determined by specifying the catch configuration $\boldsymbol{q}_\mathrm{c}$ and the catch time t_c.

In the following the three objective functions, that lead to significantly different catch behaviors, and the constraints are presented.

1) Objective Function: The first catch behavior is called "soft" mode, as it tries to make soft movements with accelerations as small as possible for each joint:

$$H_\mathrm{soft}(\boldsymbol{q},t) = \sqrt{\frac{1}{N}\sum_i^N \left(\frac{a_i}{a_{\mathrm{max},i}}\right)^2}.$$

Note, $a_i = a_i(q_i,t)$ is defined by substituting $q_i(t_\mathrm{c}) = q_i$ and $\dot{q}_i(t_\mathrm{c}) = 0$ in eq. 2 and (analytically) solving the resulting quadratic equations.

The second catch behavior is called "latest" mode. It tries to intercept the ball as late as possible on its trajectory, hence

$$H_\mathrm{latest}(\boldsymbol{q},t) = -t.$$

Finally, we implemented the "cool" mode catch behavior, where the robot first moves as fast as possible to a configuration where it can intercept the ball on its trajectory and then "cooly" waits for the ball reaching the hand and finally suddenly closes the fingers to grasp the ball. Unlike for the other catch modes, here one has to distinguish between the time t_c, the robot reaches the catch configuration $\boldsymbol{q}_\mathrm{c}$ and the time t_g, the ball is finally grasped.

$$H_\mathrm{cool}(\boldsymbol{q},t) = \sqrt[4]{\frac{1}{N}\sum_i^N t_{\mathrm{min},i}^4},$$

with $t_{\mathrm{min},i} = t_{\mathrm{min},i}(q_i)$ being the minimal time the joint i needs to reach the given angle q_i when moving as fast as possible, i.e. with $|a_i| = a_{\mathrm{max},i}$. The minimal time $t_{\mathrm{min},i}$ is defined by substituting $\dot{q}_i(t_{\mathrm{min},i}) = 0$ and $q_i(t_{\mathrm{min},i}) = q_i$ in eq. 2 and solving the resulting quadratic equation.

2) Constraints: The task – where, when and in which orientation to catch the ball – is only completely defined by means of the *equality constraints*. Let $\boldsymbol{T}_R(\boldsymbol{q}) = \left(\hat{\boldsymbol{e}}_\mathrm{x}(\boldsymbol{q}), \hat{\boldsymbol{e}}_\mathrm{y}(\boldsymbol{q}), \hat{\boldsymbol{e}}_\mathrm{z}(\boldsymbol{q}), \boldsymbol{x}_R(\boldsymbol{q})\right)$ be the robot TCP frame (see Fig. 5) and $\boldsymbol{x}_B(t)$ and $\boldsymbol{v}_B(t)$ the ball position and velocity at time t. Let further $\varphi_\mathrm{B}(\boldsymbol{q},t)$ and $\vartheta_\mathrm{B}(\boldsymbol{q},t)$ be the angles one has to rotate the vector $\boldsymbol{v}_B(t)$ first around $\hat{\boldsymbol{e}}_\mathrm{x}(\boldsymbol{q})$ and then around $\hat{\boldsymbol{e}}_\mathrm{y}(\boldsymbol{q})$, respectively, so that the resulting vector $\boldsymbol{v}'_B$

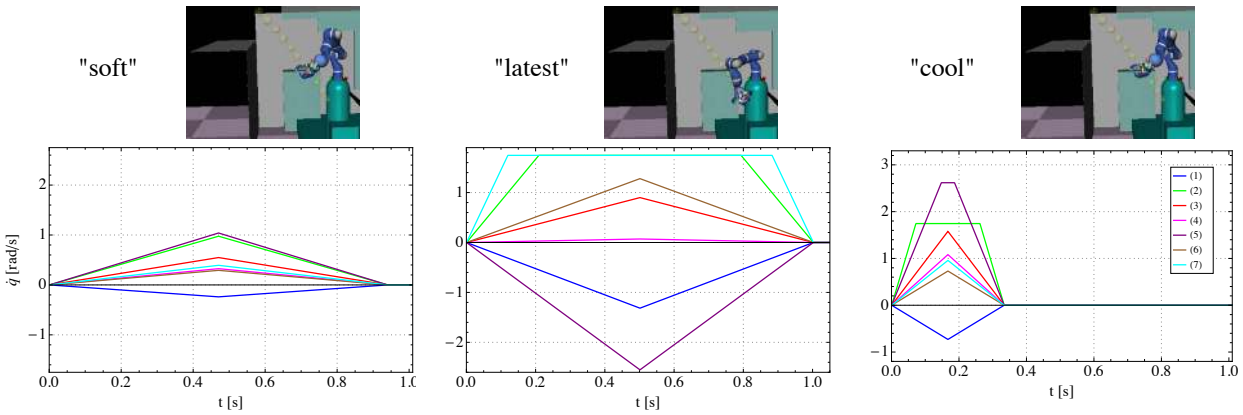


Fig. 7. Planning results for a simulated ball trajectory. For each catch behavior the first row shows the resulting catch configuration $\boldsymbol{q}_c$ and the second row the joints velocity ramps $q_i(t)$. For "latest" mode the robot catches the ball later ($t_{c,latest} = 1.0$s) on the ball trajectory and has therefore to move much faster and further, than in "soft" ($t_{c,soft} = 0.94$s) and "cool" ($t_{c,cool} = 0.33$s and $t_g = 0.94$s) mode. The catch configurations for "soft" and "cool" mode are almost the same, but the catch times differ significantly. The smoothest velocity ramps are achieved in "soft" mode, whereas the other two modes reach the acceleration and velocity limits in some joints.

becomes anti parallel to $\hat{\boldsymbol{e}}_z(\boldsymbol{q})$. The equality constraints can then be written as

$$\begin{aligned} \boldsymbol{x}_R(\boldsymbol{q}) &= \boldsymbol{x}_B(t), \\ \varphi_B(\boldsymbol{q},t) &= 0, \\ \vartheta_B(\boldsymbol{q},t) &= 0. \end{aligned}$$

The *inequality constraints* take care, that the solution $(\boldsymbol{q}_c, t_c)$ stays inside the feasible part of the solution space and are defined as follows.

- joint angle limits:

$$q_{\min,i} \leq q_i \leq q_{\max,i}, \quad i = 1 \ldots N$$

- catch time limits, with $t_{\max,c} = 1.8$s for our setup:

$$0 < t < t_{\max,c}$$

- minimal time limits, which avoid infeasible ramps, i.e. ramps which need a $a > a_{\max}$ or joint velocities $\omega > \omega_{\max}$:

$$t \geq t_{\min,i}(q_i), \quad i = 1 \ldots N,$$

  with $t_{\min,i}$ as defined in Sec. IV-B.1.
- work cell limits, where $d(\boldsymbol{x}, W)$ is the distance between a point $\boldsymbol{x}$ and one of the $N_W = 8$ workcell objects W as defined in Fig. 6:

$$d(\boldsymbol{x}_R(\boldsymbol{q}), W_k) > 0, \quad k = 1 \ldots N_W.$$

In sum, there are $N_h = 5$ equality and $N_g = 2N + 2 + N + 8 = 31$ inequality constraints.

C. Simulation Results

To visualize the differences in the three catch behaviors, for each mode the planner was run for the very same simulated ball trajectory and the very same start configuration $\boldsymbol{q}_0$ as shown in Fig. 2. The catch behavior differs significantly and is discussed in Fig. 7.

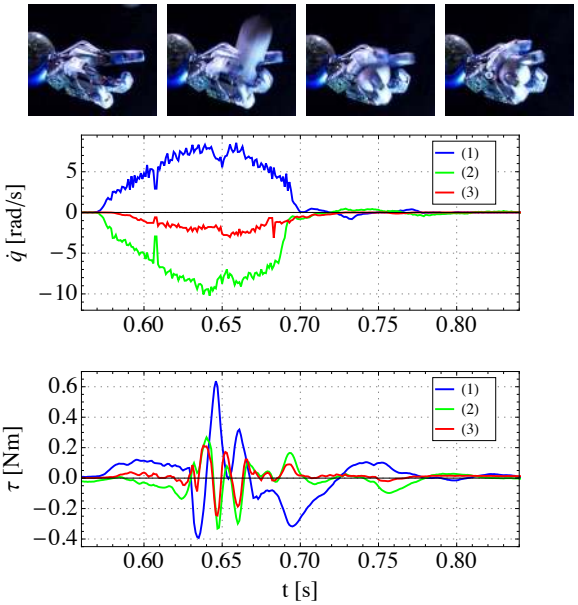
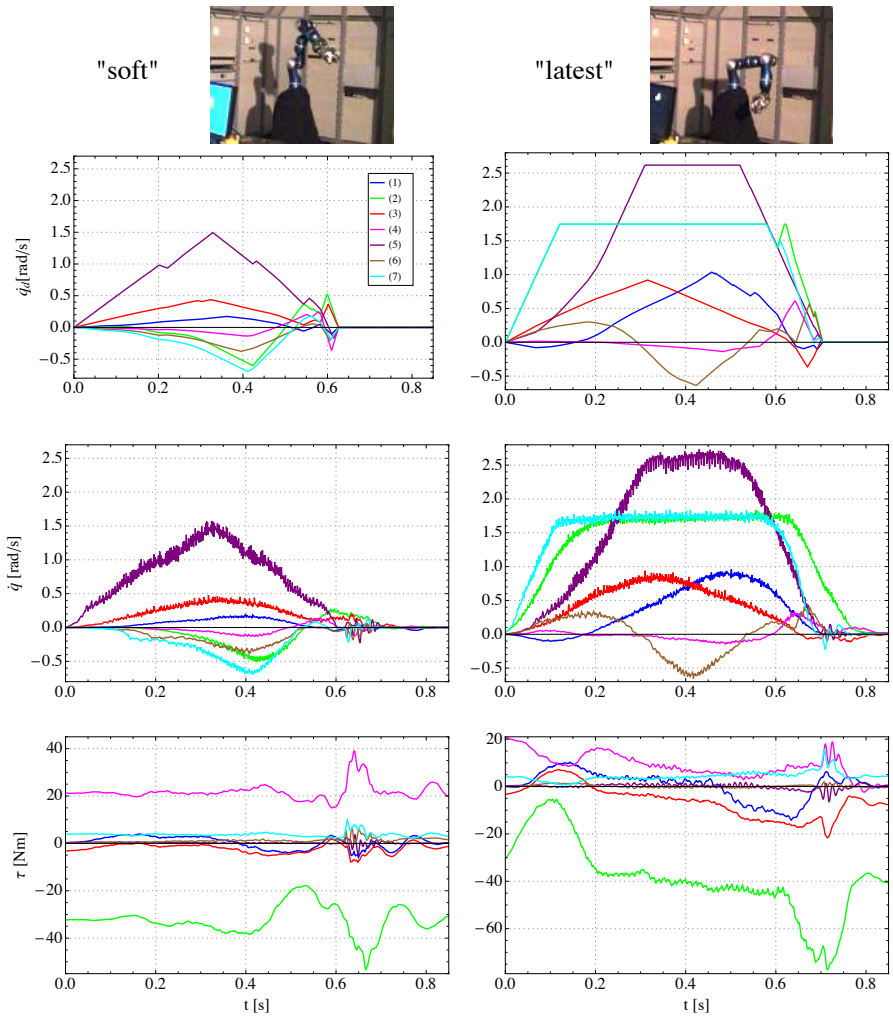


Fig. 8. First row shows the hand grasping the ball for a typical real catch ($\Delta t = 40$ms). According to the catch frame the ball approaches the index and the middle finger. At the third image of the sequence the ball is already caged but still moving. Note the fast motion of the thumb and the ring finger to block the flight channel. In the last image the stable grasp of the ball is observed. The second and third row show plots of the velocities and joint torques of the thumb (for the very same throw as the "soft" mode throw in Fig. 9). The impact at time $t_c = 0.63$ can be clearly seen as it is accompanied by a peak in velocity and impact torques that appear at the joints. The large initial peak is due to the impact while the following vibrations are also due to the short vibration phase of the arm that induce inertial forces in the fingers. In the stable grasp phase the thumb torques become very small indicating that the index and the ring finger fix the ball in the hand.



9. Experimental results for "soft" and "latest" mode catch behavior. For two almost similar ball trajectories (thrown by a human) each column depicts first the final catch configuration and then the plots of the commanded $\dot{q}_{\mathrm{d},i}(t)$ and measured $\dot{q}_i(t)$ joint velocity ramps and the measured joint torques. The catch times are $t_{\mathrm{c,soft}} = 0.63\mathrm{s}$ and $t_{\mathrm{c,latest}} = 0.70\mathrm{s}$.
A noticeable difference to the simulation results are the kinks in the commanded velocity ramps. These are due to the errors in the ball trajectory prediction: the kinks occur at the times, when the prediction (and hence the prediction error) jumps (compare Fig. 3 for the "soft" mode throw; note, that the time delay of 90ms between camera shutter time and the robot reacts to this new measurement has been compensated for; so a prediction at time t in Fig. 3 leads to robot reaction at the same time t in Fig. 9).
The executed velocity ramps follow very precisely the commanded ramps, except for a little contouring error. Even the torques roughly resemble the ideally expectable acceleration step functions, however superposed with noise and dynamical and gravitational forces the planner does not account for. This justifies our assumptions "kinematically planning" and "trapezoidal velocity ramps". Also the timing is very precise: the impact of the ball is clearly visible in the torque signals and happens at the commanded catch time.

Fig. 10. Image sequence (from upper left to lower right) for "latest" mode catching with $\Delta t = 120\mathrm{ms}$ (same throw as in Fig. 9) and $t \approx 0s$ for the first and $t \approx t_{\mathrm{c,latest}} + 20\mathrm{ms} = 0.72\mathrm{s}$ for the last image.

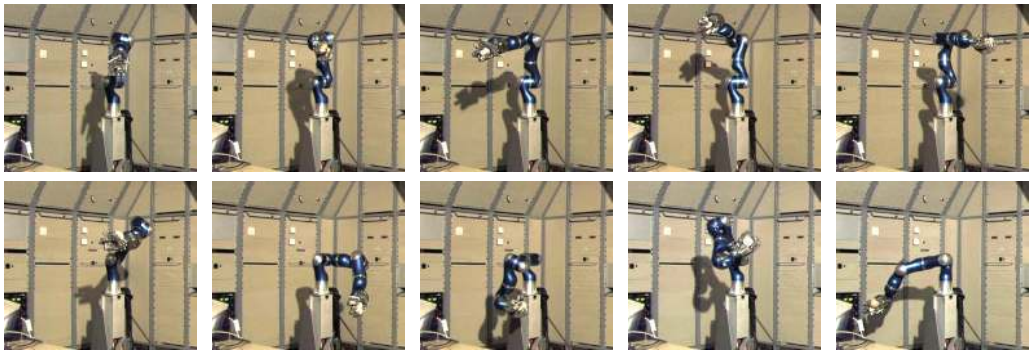


Fig. 11. Consecutive catch configurations for a continuous catch sequence without moving back to a start configuration before the next ball is thrown ("soft" mode).

V. EXPERIMENTS

In Fig. 9 we show experimental results for the "soft" and "latest" catch modes recorded for two similar (as fas as possible for the human thrower) throws. The "cool" mode is not shown, because, as discussed in Sec. IV-C, the catch configuration would be almost the same as for the "soft" mode and the distinguishing property, that the catch configuration is reached early and then the robot stays still while waiting for the ball to arrive, is corrupted due to the ever changing predictions of the ball trajectory because of tracking errors in the vision system [3].

For the "latest" mode Fig. 10 (same throw as in Fig. 9) shows an image sequence of the catch movement and Fig. 8 a closeup of the hand while grasping the ball.

To get an impression of the wide range of feasible catch configurations, in Fig 11 a sequence of consecutive catch configurations for the "soft" mode is shown, where, other than usual, the robot did not move back to its start configuration before a new throw is made, but uses the last catch configuration as its new start configuration, i.e. $\boldsymbol{q}_0^{(n+1)} = \boldsymbol{q}_c^{(n)}$. This justifies our third assumption, the collision avoidance heuristics.

The catch success rate (number balls caught versus the number of balls thrown into the feasible catch spaces) is $> 80\%$ for "soft" and "latest" mode, where the main source of failures is the vision system with its prediction errors and lost ball tracks. Additional results are contained in the the video attachment to this paper.

VI. CONCLUSION

Besides being an attraction in demos, in the first place ball catching is, due to its dynamic nature, an excellent testbed for robotic key skills and their tight interaction under hard timing constraints, where even the robotic nonspecialist can easily judge the performance of the system.

In our system currently the visual ball tracking is the limiting factor for the catch rate, whereas the mechanics and control of the arm and especially of the hand work highly reliable in grasping the ball. Also, the unified view onto the planning problem by using *online kinematically realtime optimization* proves – besides some assumptions – to generate movements which operate near the performance limits (speed and acceleration) without damaging the mechanics and can take fully advantage of the redundancy in the problem [4]. Moreover, the chosen approach allows to intuitively formulate objective functions to determine the desired catch behavior. Finally, the system architecture permits a tight interplay of the modules running in parallel on distributed computing resources under hard realtime constraints.

Currently we are substantially extending our methods to bring ball catching to our mobile humanoid robot "Rollin' Justin" [21]. This poses, mainly due to the much larger number of degrees of freedom (22 DOF including the mobile base) with much more complex dynamical couplings, a number of new, interesting challenges for all components of the system.

[3] Hence, the robot can not stay still at all until the catch time !

[4] The robot has 7 DOF, but only 4 DOF are determined, because the orientation of the catch frame around the z-axis and the catch time, when to catch the ball on its trajectory, are arbitrary

REFERENCES

[1] B. Hove and J. Slotine, "Experiments in robotic catching," in *Proceedings of the 1991 American Control Conference*, 1991, pp. 380–385.

[2] W. Hong and J. Slotine, "Experiments in hand-eye coordination using active vision," in *Proceedings of the Fourth International Symposium on Experimental Robotics, ISER95*, 1995.

[3] K. NISHIWAKI, A. KONNO, K. NAGASHIMA, M. INABA, and H. INOUE, "The humanoid saika that catches a thrown ball," in *Proceedings of the lEEE Intermtional Workshop on Robot and Human Communication*, 1997, pp. 94–99.

[4] 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. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2001.

[5] M. Riley and C. G. Atkeson, "Robot catching: Towards engaging human-humanoid interaction," *Autonomous Robots*, vol. 12, pp. 119—128, 2002.

[6] C. Smith and H. I. Christensen, "Using cots to construct a high performance robot arm," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[7] G. Hirzinger, N. Sporer, A. Albu-Schäffer, M. Hähnle, R. Krenn, A. Pascucci, and M. Schedl, "DLR's torque-controlled light weight robot III - are we reaching the technological limits now?" in *Proceedings of the IEEE/RSJ International Conference Robotics and Automation (ICRA)*, 2002, pp. 1710–1716.

[8] J. Butterfaß, M. Grebenstein, H. Liu, and G. Hirzinger, "DLR-Hand II: Next generation of a dextrous robot hand," in *Proceedings of the IEEE/RSJ International Conference Robotics and Automation (ICRA)*, 2001, pp. 109–114.

[9] QNX Software Systems. [Online]. Available: http://www.qnx.com/

[10] P. Spellucci, "A sqp method for general nonlinear programs using only equality constrained subproblems," *MATHEMATICAL PROGRAMMING*, vol. 82, pp. 413–448, 1998.

[11] ——, "Donlp2 short users guide," ftp://ftp.mathematik.tu-darmstadt.de/pub/department/software/opti/DONLP2, 1999.

[12] 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.

[13] A. Albu-Schäffer, O. Eiberger, M. Fuchs, M. Grebenstein, S. Haddadin, Ch. Ott, A. Stemmer, T. Wimböck, S. Wolf, and G. Hirzinger, "Anthropomorphic soft robotics - from torque control to variable intrinsic compliance," in *International Symposium on Robotics Research*, 2009.

[14] Ch. Ott, A. Albu-Schäffer, A. Kugi, and G. Hirzinger, "On the passivity based impedance control of flexible joint robots," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 416–429, April 2008.

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

[16] M. Görner, T. Wimböck, and G. Hirzinger, "The DLR crawler: evaluation of gaits and control of an actively compliant six-legged walking robot," *Industrial Robot: An International Journal*, vol. 36, no. 4, pp. 344–351, 2009.

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

[18] H. GONZÁLEZ-BAÑOS, 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.

[19] D. BERENSON, S. SRINIVASA, D. FERGUSON, and J. KUFFNER, "Manipulation planning on constraint manifolds," in *IEEE International Conference on Robotics and Automation (ICRA09)*, 2009.

[20] J. BETTS, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[21] C. Borst, T. Wimböck, F. Schmidt, M. Fuchs, B. Brunner, F. Zacharias, P. R. Giordano, R. Konietschke, W. Sepp, S. Fuchs, C. Rink, A. Albu-Schäffer, and G. Hirzinger, "Rollin' justin: Mobile platform with variable base," in *Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA 2009)*, 2009, pp. 1597–1598.

Acknowledgements: We thank the arm and hand development teams.