Constructing a High Performance Robot from Commercially Available Parts

Christian Smith and Henrik I Christensen
Centre for Autonomous Systems
Royal Institute of Technology
Stockholm, Sweden
Email: {ccs,hic}@kth.se

***Index Terms*— ball catching, robot design, teleoperation.**

I. Introduction

A large number of robot manipulators have been designed over the last half century, and several of these have become standard platforms for R&D efforts. The most widely used is without a doubt the Unimate PUMA 5xx series. The general availability of a platform, at a reasonable price, is important to allow design of systems that can be replicated and further developed by others. Recently there have been attempts to utilize standard platforms as exemplified by the LAGR program organized by DARPA [1]. The RobotCub project has also made a few robots available to the research community [2].

As actuation systems have become more powerful and miniaturized it has become possible to build dynamical robot systems to perform dynamic tasks. Early examples of dynamic robot control include the ping-pong playing robot at Bell Labs [3], and the juggling robot developed by Koditschek et al [4], [5]. Another example of dynamic systems are walking robots [6].

However, for research work it is often a challenge to get access to a high performance robot which is also available to other researchers. In many respects robotics has lacked standard systems based upon which comparative research could be performed. Too much research is performed on a basis that cannot be replicated, reproduced or reused. For basic manipulation there has until recently been but limited access to lightweight manipulators with good dynamics.

KUKA and DLR has announced a new manipulator that is scheduled to be on the market by late 2008, but so far the system is only marketed in Europe and the price is expected to be high. In this paper we describe design of high performance robot manipulator that is built from components of the shelf (COTS) to allow easy replication. In addition it was designed to have enough dynamics to allow ball catching, which in reality implies that the system has adequate dynamics for most tasks.

In Section II we present an application requiring significant dynamic performance and the design of a platform that fulfills the requirements. The construction of the platform is described in Section III, and in Section IV we present our first experimental evaluation. A photo of the final implementation is shown in Fig. 1.

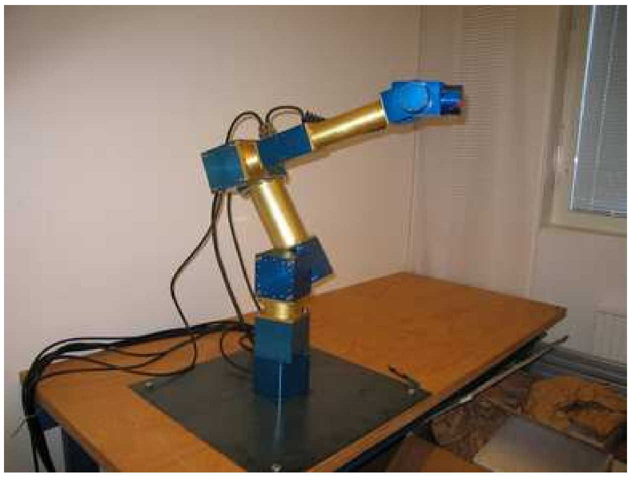


Fig. 1. The high-performance manipulator.

II. Design Procedure

This section provides an initial analysis of the requirements for a system to perform teleoperated ball-catching. A design for the system is developed from the analysis of requirements. The performance of the design is verified in simulation.

A. Experimental Requirements

The main type of experiments that we want to perform involve catching a ball thrown across a room. We anticipate a normal, slow, underhand throw from a distance of approximately 5 m. In an indoor environment, a ball can be thrown with reasonable accuracy along a parabolic path with an apex of 2.3 m, with both the thrower and the catcher situated at a height of approximately 1 m (see Figure 2). Simple studies of human performance indicates that the system must be able to accommodate variations in thrower accuracy corresponding to catching the ball within a 60$\times$60 cm window. From these basic requirements it is possible to compute flight time and velocities for the scenario, as summarized below:

- Throwing distance will be approximately 5 m.
- Flight time will be up to 1 s, the typical time is expected to be 0.8 s.
- The ball will travel with an approximate velocity of 6 m/s at time of arrival.

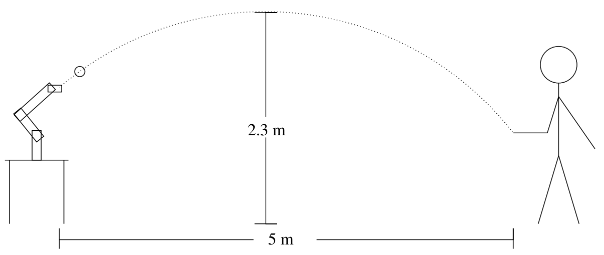


Fig. 2. Schematic of ballcatching experiment.

- The ball should be caught if it arrives within a 0.6 m $\times$ 0.6 m window.

B. Platform Requirements

The experimental requirements stated in the previous section impose requirements on the platform. One desired feature is the use of standard video cameras for trajectory estimation. With normal 50 Hz cameras, the frame time is approximately 20 ms, and a similar time window is expected to be needed for segmentation and position estimation. In addition, at least three frames are required for trajectory estimation, but limited accuracy of the cameras will mean that more, probably as many as 10 images might be necessary (c.f. [7]). Thus, the time delay from the initiation of a throw to the initial trajectory estimate might be 200 ms. This particular setup is also intended to be used for teleoperated catching, were a human operator steers the robot towards the predicted trajectory, so we have to allow for the operator's reaction time as well. This might be around 100 ms, so a window of 300 ms is reserved for initial reaction to a throw, leaving 500 ms in which the arm has to move into position. In the worst-case scenario, the arm has to move against gravity from one opposing corner of the operational window to another, a distance of almost 0.9 m. Since the initial experiments will not be concerned with grasping, a simple passive end effector like a small bucket will be employed, and the positioning error has to be smaller than the radius of the bucket, preferably less than 1 cm. These requirements can be summarized as:

- End effector has to be able to move 0.9 m in 0.5 s, (partially) against gravity, from stand-still to stand-still.
- The precision of positioning the end effector should be within 1 cm

Given constant acceleration and deceleration, a distance of 0.9 m can be traveled in 0.5 s if the acceleration is at least 14.4 m/s^2, and the maximum velocity is at least 3.6 m/s. This also has to be achieved when working against gravity. These are the minimum dynamic requirements — the actual implementation should have some margin to allow for uncertainties.

To enable flexibility in the design of future experiments, it is desirable to allow for different types of sensors to be mounted in the end effector reference frame, so this should be freely orientable in a dexterous manner. The end effector therefore has to have 6 degrees of freedom, and should be freely orientable within the entire operation window.

The system thus requires significant dynamics and the control has to be performed in real-time. This implies that it is desirable to have closed form solutions for kinematics, which in turn imposes constraints on the design of the overall kinematic structure. Without a closed form kinematic/dynamic solution it would be much more challenging to guarantee the real-time performance.

A highly dynamic robot arm will pose a potential hazard to both its operator and itself unless sufficient precautions are taken. Therefore, the control of the arm has to be sufficiently exact so that safe paths can be accurately followed, and precautions against malfunctions have to be taken. The former requires control loops running at a high frequency/low latency, the latter that software and hardware malfunctions are kept at a minimum, and that the negative effects of malfunctions should be minimized. Thus, the software environment has to be a stable real-time system, while the hardware contains fail-safe fallback for dealing with software failure.

These further requirements a solution imposed on the design are summarized below:

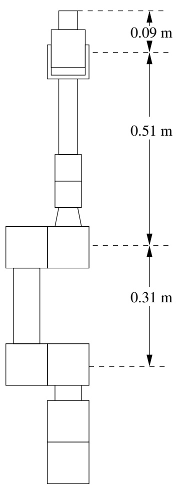
- Closed form analytical kinematics and dynamics are necessary for speed.
- At least 6 degrees of freedom.
- Acceleration of at least 14.4 m/s^2 for end effector.
- Velocity of end effector of at least 3.6 m/s.
- Safety for operator and machinery requires a stable real-time system, as well as fault-tolerant hardware.

C. Designed Solution

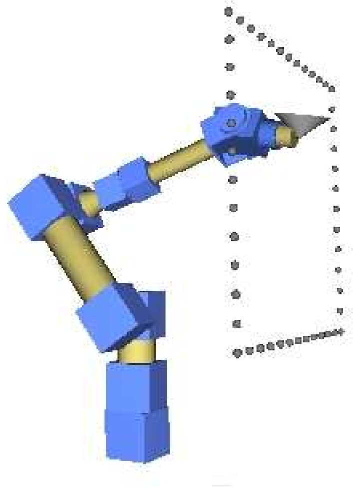
There are a number of fairly fast robotic manipulators commercially available, like for instance the Kuka Light Weight Arm [8]. It has been shown to be fast enough to catch thrown balls autonomously [7], but needs a very early ballistic path estimate to be able to do this. In our experiments we also want to include a human operator in the control loop to do semi-autonomous teleoperated catching, so we require even faster movements to compensate for slow human reactions. With perhaps only half the time to get into position, twice the speed is needed.

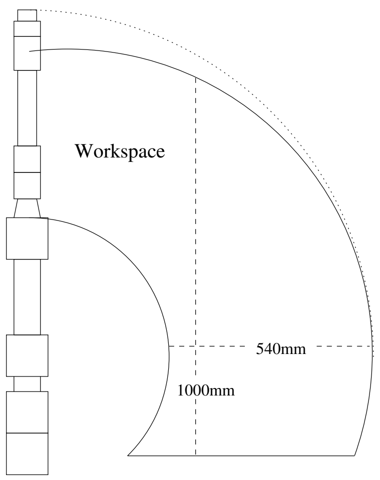
In order to cater to the special needs of our experiments, we decided to construct our own 6 DoF arm, and to examine if this could be done using *PowerCube* modules from Amtec. These modules are available off the shelf and allow rapid prototyping. The range of modules clearly include some that have specifications which are adequate for the target application (See Section III-A). These modules also have a built-in controller that can be used for embedded safety functions.

The actual performance depends on the configuration that the modules are assembled in, so a few different configurations were examined more closely in computer simulation, where a 10 % uncertainty was added to the maker specifications. The configuration that showed the most promising results is one that is kinematically very similar to a Puma560 arm (and to many other commercially available robots). This is not only a configuration that allows for very good dynamic performance (see Section II-D), but as it has been widely used and studied, several implementation issues are already solved, thus making the design process considerably faster. For example, the closed form solution for inverse kinematics



(a) Dimensions


(b) 3D rendering of arm and operational window



(c) Workspace with tool oriented towards user.

Fig. 3. The first design of the manipulator, using Amtec PowerCubes.

and dynamics are well-known. Keeping the moments of inertia as low as possible in the moving parts, and placing heavier, more powerful modules where their impact on the inertial load is lower, very fast dynamics can be achieved. In the final design, three 1.5 kW motors are used to position moving parts weighing approximately 10 kg. Also, the arm is designed so that the center of mass of the moving parts will be close to the rotational axis of the first joint when working in the intended window of operation. This will balance the system and keep down the strains on the first joint.

The choice of gearings and link lengths induce a trade-off between acceleration and end effector velocities. The balancing of this trade-off has been made to minimize the time needed to move the end effector from one stationary position to another within the operation window. Since there is a limited, discrete amount of possible combinations of actuators, it was possible to find the optimum through an exhaustive search. The resulting configuration that performed the best in simulation is specified in Table II. The design and dimensions can be seen in Figure 3. The design allows for a workspace that is more than large enough to accommodate for the specified 60 cm $\times$ 60 cm window for ballcatching, though the manipulator's dynamic performance deteriorates somewhat at the edges of the workspace. A cross-section of the workspace can be seen in Fig. 3(c). The arm has rotational symmetry as viewed from above, but is for safety reasons limited to a half-circle to avoid collisions with any objects behind it.

The control setup for the manipulator should be a realtime system with as short a looptime as possible. The PowerCube modules support several different communication protocols, but for robustness and responsiveness, the option of a 1 Mbit/s CAN bus was deemed optimal. The hardware design can be implemented in several different ways. In principle all modules could be on a single CAN bus or each module could have a bus of its own. The end-effector joint is a combined pan-tilt which requires use of a single bus to control both degrees of freedom. Depending on the number of modules per bus, the lengths of the control cycle will vary (see Section IV-C). This means that the control computer could be equipped with either 1, 2, or 3 CAN controllers for symmetric loads, or 4 or 5 controllers for assymetric loads, where the inner joints that control positioning are run at a higher frequency than the outermost controlling orientation. Simulations where the inner joints were controlled at 500 Hz and the outer joints at 200 Hz show that this is a viable option. In simulation, the inner joints can be stably controlled at full power output using frequencies from approximately 400 Hz and upwards, but the real world implementation may have slightly different requirements.

The first choice for the computer performing the direct low-level control of the robot was RTAI, a real time GNU/Linux system that has showed good performance in previous studies [9], but some testing led to the choice of GNU/Linux patched to high resolution timers[1], as this not only showed better realtime performance but allows for easier implementations in user space. The control computer will also perform the trajectory generation and be responsible for dynamic and kinematic calculations.

Teleoperation should be enabled by allowing connection to an external computer that runs the user interface (UI). The communication to the UI computer should be in cartesian space, since the kinematic structure of the arm allows for up to eight different joint-space configurations for each cartesian position, and the choice of configuration should be made locally by the real-time low-level controller for best performance. The connection is made over UDP/IP, as this has been shown to give significantly better control performance than TCP/IP over an internet connection (see e.g [10] for a complete evaluation). The connection to the UI will not need hard realtime performance, but the smaller the time lag can be made, the better the performance. In early experiments over

[1]`http://www.tglx.de/hrtimers.html`

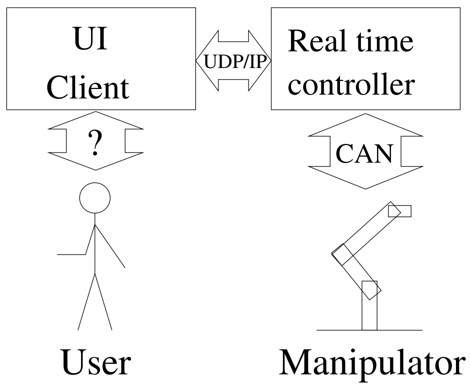


Fig. 4. Schematic of the connection architecture for a teleoperation scenario.

the LAN in our lab, the total roundtrip time from the UI input via the manipulator controller to UI feedback has been shown to be in the range of 10–20 ms. A schematic of the connection architecture is shown in Figure 4.

1) Specifications: The basic design specification is outlined below:

- 6 DoF arm made with Amtec PowerCubes
- Kinematic configuration of *Puma560* type
- GNU/Linux, preferably with high resolution timers, for control computer
- Communication over several parallel CAN connections.

D. Simulated Performance

The performance of the proposed arm was first calculated using the specifications from Amtec and numerical simulations. The results for the travel times are of course dependent on the type of controller used, and in this particular case, the controller included a dynamic model for feed-forward control, and the torques in each individual joint were set in order to achieve a target velocity as quick as possible. The target velocity was chosen as the minimum of the actuators maximum velocity and the highest velocity from which stopping at the desired position was achievable. This latter factor was calculated using the maximum torque of the actuators and the inertial load of the current configuration, a figure that was multiplied with a factor slightly less than 1 to achieve a margin. Using this simple controller, the simulated arm had more than adequate performance, as is summarized in Table I. Note that the first acceleration figure given is the maximum achievable for small movements, and the second figure for larger, cross-workspace movements.

III. Implementation

This section describes the technical details of the actual implementation of the robot arm.

A. Hardware implementation

The arm proposed and specified in the earlier sections was constructed and mounted on a sturdy industrial work table (see photo in Figure 1). The lower three actuators have a maximum output of almost 1.5 kW each, harmonic drive gearboxes and incorporated brakes to lessen motor strain when not moving. The fourth actuator is similar, but considerably smaller as it carries a lighter inertial load. The maximum output is 0.36 kW. The last two joints are contained in a combined pan/tilt unit. This is less powerful, but has lower weight per joint than other solutions. This also incorporates the same gearbox and brakes as the other modules. Specifications can be found in Tables II, III and IV

TABLE I
Simulated performance of robot arm

Since performance is highly dependent of arm position, all values are given as their lower limit within the window, and are thus equal to or better than this in the entire window. The first acceleration given is the maximum possible for small movements, the second (in braces) is the maximum acceleration achievable for large movements.

Endpoint acceleration	$> 100\ m/s^2$	$(> 30 m/s^2)$
Endpoint velocity	$> 5\ m/s$	
Traveltime across window vertically, from standstill to standstill	$< 0.36\ s$	
Traveltime across window diagonal, from standstill to standstill	$< 0.37\ s$	
Traveltime from window center to upper corner, from standstill to standstill	$< 0.22\ s$	
Repeatability of position	$\pm 1\ mm$	

TABLE II
Specifications for the parts used in the manipulator.

Part	Product name	mass	Comment
1st joint	PowerCube PR110	5.6 kg	51:1 reduction gear
1st link	PAM104	0.2 kg	55mm cylindrical rigid link
2nd joint	PowerCube PR110	5.6 kg	101:1 reduction gear
2nd link	PAM108	0.8 kg	200mm cylindrical rigid link
3rd joint	PowerCube PR110	5.6 kg	51:1 reduction gear
3rd link	PAM119	0.2 kg	45mm conical rigid link
4th joint	PowerCube PR070	1.7 kg	51:1 reduction gear
4th link	PAM106	0.6 kg	200mm cylindrical rigid link
5th,6th joint	PowerCube PW070	1.8 kg	2DoF wrist joint

TABLE III
Manufacturer's specifications for the joint actuators.

Joint no.	max torque	max angular velocity	repeatability
1	134 Nm	8.2 rad/s (470^o/s)	±0.00035 rad
2	267 Nm	4.1 rad/s (238^o/s)	±0.00035 rad
3	134 Nm	8.2 rad/s (470^o/s)	±0.00035 rad
4	23 Nm	8.2 rad/s (470^o/s)	±0.00035 rad
5	35 Nm	4.3 rad/s (248^o/s)	±0.00035 rad
6	8 Nm	6.2 rad/s (356^o/s)	±0.00035 rad

The PowerCube modules have a simple onboard controller that implements basic security features. They will not allow motion beyond certain angle limits (that can be set by the user), and will perform an emergency stop if these limits are

TABLE IV
THE DENAVIT-HARTENBERG PARAMETERS FOR THE ARM, USING J.J. CRAIG'S NOTATION.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	$0°$	0 m	0 m	θ_1
2	$-90°$	0 m	0 m	θ_2
3	$0°$	0.31 m	0 m	θ_3
4	$-90°$	0 m	0.51 m	θ_4
5	$-90°$	0 m	0 m	θ_5
6	$90°$	0 m	0 m	θ_6

exceeded, or if no watchdog signal has been transmitted over the CAN bus for 50 ms.

In order to avoid collisions, both with itself and environment, the angles of the different joints have been limited to the values shown in Table V. There are two sets of limits, each set prohibiting collisions in itself but with a limited workspace. The system will switch limit sets when moving out of range of one set and into range of another, with an intermediate limit set that consists of the tighter limits of the two sets. This means that even if communication would break down in the middle of a limit switch, the individual modules will always be limited to safe intervals, while at the same time allowing for use of a large part of the robot's potential workspace.

To test these safety measures, two experiments were conducted. In the first experiment, the communication link was severed between the computer and the robot. This results in a termination of the watchdog update, and the modules finish their last command and engage the brakes. In the second security experiment, illegal position commands were intently issued by the control program. The modules' onboard controller correctly identified these as violating joint limits. The arm moved into the legal position that was closest to the commanded position and stopped. This should account for safe handling of an unexpected breakdown of control algorithms, the control computer, or the CAN communication link.

TABLE V
LIMITS ON JOINT ANGLES.

Joint no	set 1	set 2
1	-90^o – $+90^o$	-90^o – $+90^o$
2	-100^o – -40^o	-130^o – -70^o
3	-60^o – 50^o	-40^o – 90^o
4	-160^o – $+160^o$	-160^o – $+160^o$
5	-120^o – $+120^o$	-120^o – $+120^o$
6	-180^o – $+180^o$	-180^o – $+180^o$

A power supply unit capable of delivering the required 30A @48V to each module was constructed using PBA-1500F power converters from Cosel. An emergency stop controller that acts directly on cutting the power was implemented so that the power unit cannot be activated without the emergency controller present. The emergency stop has been tested and works well, as a power cut will stop the modules and engage the brakes.

The communication interface was designed to be implemented over 4 separate CAN buses, one each for the 3 inner (position controlling) joints, and one common bus for the 3 outer (orientation controlling) joints. Two 2-channel PCI CAN controllers from Kvaser were chosen, as these had open source device drivers for Linux that where deemed plausible to port to real-time usage.

A Dell PowerEdge 1800 server with a 3.6 GHz Intel Xeon processor was acquired to use as control unit. This choice was based upon a balance of requirements for processing power and reliability.

B. Software implementation

A Linux 2.6.19 Kernel was patched with high resolution timers for low latency realtime performance. A customized communications API was implemented to guarantee low-latency communication with the PowerCube modules, as well as customized libraries for fast vector manipulations optimized for calculating arm dynamics. The control loop is run in soft real-time. Experiments have shown that this gives a worst case latency of less than 100 μs, which is sufficient. The average jitter for the main loop of the control algorithm is 6 μs, which is significantly less than the modules' latency of up to 600 μs. This soft real-time performance is comparable to that of hard real-time systems like RTAI, but with the advantage of simple straight-forward userspace implementation.

Inverse kinematics and dynamics are calculated using a C implementation of the analytical solution for a Puma arm in [11], and the forward dynamics are calculated using the second algorithm in [12]. As a result, inverse kinematics can be calculated in 1.7μs, and dynamics in 41μs, so that all calculations needed in the control loop take less than 50μs. This means that virtually all latency in the control loop originates from the CAN bus communication path and the PowerCube modules response times.

Combined position and velocity control has been implemented on the system using a combined feed-forward computed torque control (CTC) scheme and a feed-back PI controller. When a new setpoint enters the controller, a velocity ramp trajectory is calculated in joint space. This trajectory is limited by a preset top velocity (presently 4 rad/s) and a maximum acceleration, but otherwise is the shortest path towards reaching the desired position θ_d and velocity $\dot{\theta}_d$ from the actual position θ_a, without exceeding the maximum allowable acceleration a_{max}, see Equation 1. This ramp works under the assumption that the desired position is frequently updated to new positions in accordance with the desired velocity.

$$\dot{\theta}_{ramp} = \sqrt{|\theta_d - \theta_a| \cdot a_{max}} \cdot sign(\theta_d - \theta_a) + \dot{\theta}_d \quad (1)$$

The maximum acceleration a_{max} is limited by a preset limit value[2] and the maximum achievable acceleration, computed by calculating the resulting acceleration with maximum torque and taking away a small safety margin. The desired acceleration fed to the the CTC controller is then the acceleration

[2]The limits on velocity, acceleration, and jerk are chosen to limit the mechanical stress on the system, while still being able to reach a given point in the workspace in less than 0.5 s.

necessary to achieve the target velocity $\dot{\theta}_{ramp}$ as soon as possible, without violating the limits on acceleration or jerk. For evaluation purposes, the acceleration has been limited to 16 rad/s^2, and jerk to 400 rad/s^3.

The ramp trajectory is recalculated in each iteration of the control loop. The current position and velocity, and the acceleration prescribed by the ramp, are fed to the inverse dynamics function that determines the necessary torques to follow the trajectory. These torques are converted to currents and sent to the actuator modules.

A corrective term consisting of a PI controller monitors the difference between desired velocity and actual velocity, and corrects the controller current accordingly. This term is necessary as the feedforward CTC controller does not contain an accurate enough model of friction, the movements of the power cords or the nonlinearities in the current/torque relationship. For a schematic of the control scheme, see Fig 5.

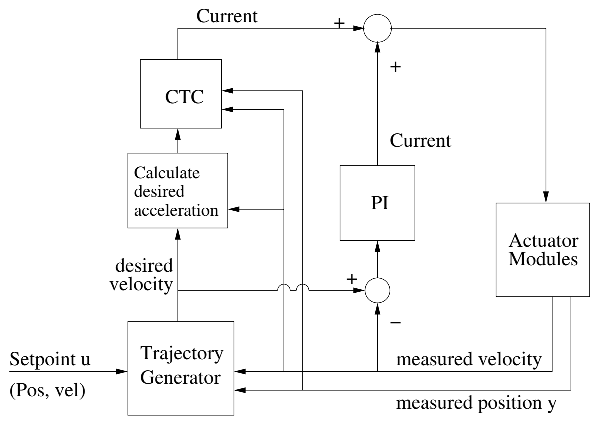


Fig. 5. Schematic of controller

IV. Performance

There is still some fine tuning remaining to be done for the robot arm, but even so, it already fulfills all the specified requirements that can be measured, and has a performance similar to the simulation.

A. Precision

In order to measure the repeatability of positioning of the arm, a paper "target" with a millimeter scale was fixed to the last joint of the arm. The arm was stopped in a position in the center of the workspace. A laser pointer capable of producing a light point approximately 1 mm in diameter was fixed to point to the center of the target. The arm was then taken around a complicated path traversing and circling the workspace for approximately one minute. The arm was then sent back to the original position. To the precision of the scale and the observer's perception, the pointer was in the middle of the target. This was repeated for several different positions and angles, with the laser pointer mounted both horizontally and vertically, with the same results. The repeatability is therefore deemed to be better than ± 1 mm. The arm has also been tested to follow a straight path with sub-millimeter accuracy, but this has only been performed at very low speeds for safety reasons, so there are no experimental results for the accuracy at higher velocities.

B. Dynamic performance

The arm has been timed to traverse the operational window shown in Figure 3 vertically (distance 60 cm) in both directions within 0.39 s from standstill to standstill, which was predicted in the simulations. As for other movements — horizontal (60 cm) and diagonal (90 cm) traversion — only times of 0.5 s have been verified, as this is enough for our application and we want to minimize mechanical stress on the equipment. However, this implies that any point-to-point motion in the operational window takes at most 0.5 s to execute. The outermost joints are slightly slower then the inner ones, so the final angular alignment of the end effector rather than the positioning is the limiting factor for many configurations.

C. Control loop times

Although the PowerCube modules are specified by the manufacturer to handle CAN bus communication up to 1 Mbit/s, experiments showed that this rate can not be maintained continuously. Especially when controlling several modules on a single CAN bus, there is a tendency for CPU overload/overheat in the modules. This results in an unrecoverable error that requires a shutdown and cooldown before operation can be resumed. The communication frequencies anticipated from the specifications can be seen in Table VI). The time to complete a communication loop consists of the 0.134 ms needed to send a CAN message at 1 Mbit/s (or 0.268 ms at 500 kbit/s), and the approximately 0.25 ms a module needs to respond to a request. The response time is somewhat dependent on the nature of the request. When performing several read/writes over the same bus to different modules, the time spent waiting for one module's response can to some extent be used to communicate with another module, hence the slight nonlinearity of loop times as a function of number of connected modules. The tables show two different speeds for each setup — with or without velocity polling. The modules have internal velocity measurements that are more accurate than just differentiating two position measurements. On the other hand, if these velocity measurements are used, the temporal resolution will be lower due to the extra time needed for communicating this additional data. Experiments have yet to show which strategy will yield the best overall performance.

In the implementation, a control loop frequency of 600 Hz for the inner 3 cubes and 200 Hz for the outer 3 cubes was used. This is with velocity polling, at 1 Mbit/s, and using one CAN bus per card for the inner cubes, and a joint bus for the outer ones. The lower frequency is obtained by only communicating with one of the outer cubes in each iteration of the control loop. Due to their limited inertia and power, the outer cubes have a very limited influence on the overall dynamic performance of the arm, and thus the error induced by scarce measurements from the outer cubes is negligible.

TABLE VI
THEORETICAL CONTROL LOOP SPEEDS OVER THE CAN BUS

	Modules per CAN controller card			
	1	2	3	6
Cycle periods at 1 Mbit/s				
with velocity polling	1.04 ms	1.30 ms	1.87 ms	3.22 ms
without velocity polling	0.52 ms	0.65 ms	0.8 ms	1.61 ms
Cycle periods at 500 kbit/s				
with velocity polling	1.57 ms	2.14 ms	3.22 ms	6.43 ms
without velocity polling	0.79 ms	1.07 ms	1.61 ms	3.22 ms
Control freq. at 1 Mbit/s				
with velocity polling	961 Hz	769 Hz	535 Hz	311 Hz
without velocity polling	1923 Hz	1538 Hz	1250 Hz	621 Hz
Control freq. at 500 kbit/s				
with velocity polling	637 Hz	467 Hz	311 Hz	156 Hz
without velocity polling	1265 Hz	935 Hz	621 Hz	311 Hz

The communication frequency is approximately 37.5% lower than the theoretical maximum, but this frequency allows the control loop to run for hours uninterupted without overheat. Also, since the lower-than-specified frequency is accomplished by padding the loop, the padding also absorbs the variations in module response time, resulting in virtually no variations in loop cycle times.

D. Calibration Loss

When using the robot for our ball catching experiments, the movements are fast, but they are centered in the designed work space (the square operational window shown in Fig II-C), and the typical duration in time is not very long. In teleoperation experiments, we have repeatedly let users move the manipulator freely for durations of up to 30 minutes. In these longer, freer sessions, we have noticed a tendency for the encoders in the joints to lose calibration. The faster the movement is, and the farther it is from the designated workspace, the larger this tendency is.

Some simple experiments were performed to verify this behavior. When only moving within the designed workspace, there was no measurable loss of calibration, even for movements at full capacity. For motions far outside the workspace, there was no measurable loss when moving at angular accelerations below 3 rad/s^2. With higher acceleration settings when moving outside the work space, loss of up to a few degress of calibration has been observed in the three lower modules. This is especially found in irregular motions. Recalibration of the encoders is a simple matter that only requires the manipulator to be returned to a predefined "home" position, but this disrupts whatever other motion was being performed.

V. Ball Catching Experiments

In order to verify the performance of the manipulator, a setup allowing for an autonomous ball catching scenario was constructed. These experiments are still at an early stage, but the early results are promising.

A. Control Server

A first prototype server application has been implemented. It receives cartesian coordinates from a client computer over an UDP/IP connection and tracks these coordinates as closely as possible, while returning information on present position and velocity in both cartesian and joint space. All commands and measurements are time-stamped in order to enable correction for time-lags over the communication link.

B. Experimental Setup

The manipulator was fitted with an end effector consisting of a passively damped cardboard cylinder with a diameter of 14 cm (see Fig. 6). We launched soft juggling balls from a distance of approximately 4 m. To ensure repeatability, the ball was launched using a mechanical launcher with a precision of ± 10 cm for this distance (see Fig 7). The ball has to hit within 4 cm of the center of the cylinder in order to be caught.

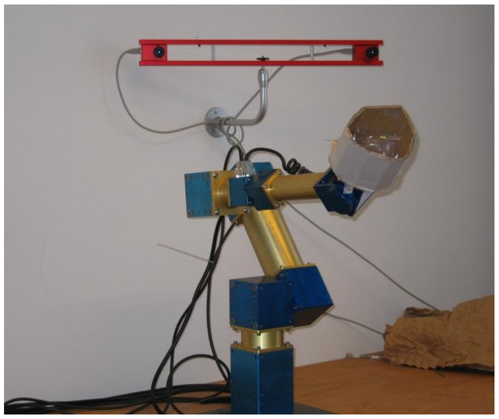


Fig. 6. The manipulator with cameras and ball-catching end effector.

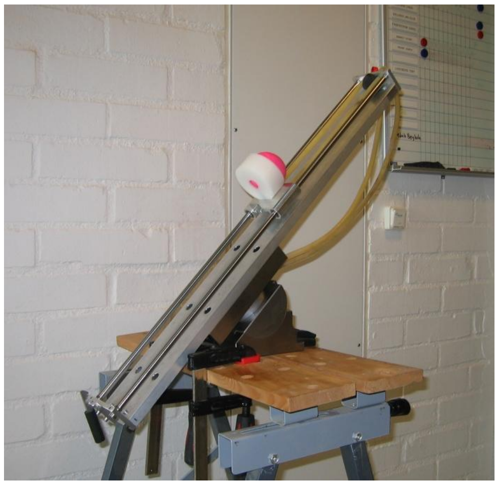


Fig. 7. The mechanical ball launcher

Using this setup, the flight time of the ball was approximately 0.8 s. The ball position was measured with a stereo vision system consisting of two Firewire cameras mounted on a 60 cm baseline approximately 0.5 m behind and slightly above the robot (see Fig 6). The ball tracking was done by using an extended Kalman filter (EKF), as described in [7].

The ball was detected in each image using simple color segmentation. First, the 24 bit RGB image was converted to

24 bit HSV using a lookup table. The ball was found to have a *hue* value of 3, and a (largely varying) *saturation* value of approximately 160, so all pixels that were in the range 1–5 for hue and 120–200 for saturation were preliminary marked as ball pixels. A second pass that only kept marked pixels with at least 3 other marked neighbors eliminated noise. The center of mass for the marked pixels was calculated and used as the ball centroid. A subwindowing scheme was applied, since these have shown to be very efficient to significantly reduce the time needed in segmenting moving objects (c.f. [13]). After the ball has been detected the first time, only a subwindow where the ball should be expected to be found was processed. This subwindow was calculated using the state estimate from the EKF, and the size of the window is set to cover several times the standard deviation in position. Using this approach, the ball can be segmented and localized with a reasonable accuracy at less than 4 ms processing time per stereo image pair, giving sufficient real-time performance.

The actual catching position was decided by interpolating the point were the predicted ball trajectory intersects the plane of the robot's workspace. This position was then sent via the UDP/IP connection to the control computer, that sent the manipulator to the position. Launching 108 balls that hit within the operating window, with an average distance of 24 cm from the manipulator's starting position, 73% were caught, 19% bounced off the rim of the end effector, and 8% were missed. The main cause of missed catches was errors in the early predictions of the ball path, causing the robot to start motion in the wrong direction.

VI. Conclusion

In this article we have presented the requirement for a highly dynamic robotic system to be used in studies for ball-catching. From these requirements and a number of secondary goals a system has been designed using off-the-shelf actuation modules. Associated software for real-time control has been designed and implemented on a commercially available computer platform. The system operates at 600 Hz and satisfies all the requirements specified for the design. Results from early experiments demonstrate that the system fulfills the static and dynamic requirements to allow ball catching.

Appendix

A. Cost Breakdown

The total cost of hardware used in the setup described in this paper was just below 50 000 euros. For a detailed cost breakdown, see Table VII. Please note that these are the actual prices paid, and that there is no guarantee for future availability at these same prices.

B. Comparison to Alternatives

Table VIII shows a summary of alternative manipulators in more or less the same performance and/or price segment. Performance figures are taken from manuals provided by the manufacturers. Prices are either qoutes or actual paid prices. It should be noted that pricing may vary significantly.

TABLE VII
Prices (in Euro) for the setup used in the present paper

Part(s)	Price (€)
Actuators	38,000
Rigid Links	3,400
CAN System	1,600
Mountings	500
Power Supply	4,400
Control Computer	1,600
Total	49,500

The proposed manipulator compares well to other options. The KR5 may have a more attractive performance/price ratio, but the proprietary interface limits control to high-level position or velocity control at 80 Hz with no low-level interface, meaning that it may not be suitable for some research applications. In contrast, the KUKA LBR has an accessible interface and torque sensors in all joints [8], but is substantially more expensive.

Different manufacturers provide different performance metrics, especially concerning velocity wich is given in either joint or cartesian space, making comparison less straight-forward. Power-to-mass ratios may give an indication of performance, but it is also worth to bear in mind that power also inversly correlates to safety, and that less powerful models may often be better suited for operation in close human proximity.

C. Other Applications

Apart from the automated ballcatching task described in Section V-B, the manipulator has been also been succesfully applied to other tasks, such as teleoperated ball catching, robot control using a Wiimote video game controller, and positioning visual targets used for automated camera calibration. Since the platform was designed for a task requiring high velocities, it has adequate performance for tasks that require lower velocities as well. The strength of the setup has been shown to be the ease at with which it can be applied to new tasks, given the completely open interface. An obvious weakness is the high power consumption, making it unsuitable for mobile applications in spite of the relaitively low weight. Planned future applications include adding force and torque sensors to enable teleoperated force control.

D. Project Webpage

As part of the effort to increase the availability of the proposed platform, there is a webpage with information regarding the platform, as well as downloadable media and source code at `www.cas.kth.se/~ccs/Robot_arm`.

Acknowledgment

The research presented here is partly funded by the 6th EU Framework Program, FP6-IST-001917, project name Neurobotics.

TABLE VIII

COMPARISON TO SOME ALTERNATIVE MANIPULATORS. DATA AS GIVEN IN MANUFACTURERS' DOCUMENTATION.

name:	price(€)[a]:	dof:	reach:	weight:	power(peak)[b]:	payload:	API:	velocity:
Puma 560	—[c]	6	0.86m	63 kg	1.5 kW	2.5kg	RT joint	0.5 m/s
Neuronics Katana	20 000	5	60 cm	4.3 kg	96 W	0.4 kg	RT traj/joint	90 deg/sec
KUKA KR5 850	22 000	6	0.85m	29 kg	2.3 kW	5 kg	80 Hz pos/vel	250 deg/s
Schunk LWA3	45 000	7	—[d]	≈10 kg[d]	0.48 kW	5 kg	joint traj/current	70 deg/s
Proposed Manipulator	50 000	6	91 cm	23 kg	5.5 kW	—[e]	600 Hz position/velocity	7m/s
Barret WAM	70 000	7	1 m	27 kg	250 W	3 kg	500 Hz traj/force	1 m/s
KUKA LBR	120 000	6	0.94m	14 kg	720 W	14 kg	1 kHz torque/force/traj	120 deg/s

(a) Actual prices paid or as quoted by supplier.
(b) Rated power consumption.
(c) Used, price varies with condition.
(d) Available in different configurations.
(e) Not tested for durability.

REFERENCES

[1] L. D. Jackel, E. Krotkov, M. Perschbacher, J. Pippine, and C. Sullivan, "The DARPA LAGR program: Goals, challenges, methodology, and phase I results," *Journal of Field Robotics*, vol. 23, no. 11–12, pp. 945–973, 2006.

[2] G. Sandini, G. Metta, and D. Vernon, "The icub cognitive humanoid robot: An open system research platform for enactive cognition," in *Artificial Intelligence 50 years*. Berlin: Springer Verlag, 2008.

[3] R. Andersson, "Understanding and applying a robot ping-pong player's expertise," in *Proc. 1989 IEEE Intl. Conf. on Robotics and Automation*, Scottsdale, AZ, 1989, pp. 1284–1289.

[4] M. Buhler and D. E. Koditschek, "From stable to chaotic juggling: Theory, simulation, and experiments," in *Proceedings 1990 IEEE Int Conf Robotics and Automation*, Cincinatti, OH, 1990, pp. 1976–1981.

[5] A. A. Rizzi and D. E. Koditschek, "Progress in spatial robot juggling," in *Proceedings 1992 IEEE Int Conf Robotics and Automation*, Nice, France, 1992, pp. 775–780.

[6] M. H. Raibert, *Legged robots that balance*. Cambridge, MA: MIT Press, 1986.

[7] U. Frese, B. Bauml, S. Haidacher, G. Schreiber, I. Schaefer, M. Hahnle, and G. Hirzinger, "Off-the-shelf vision for a robotic ball catcher," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, pp. 1623–1629.

[8] G. Hirzinger, N. Sporer, A. Albu-Schafer, M. Haahnle, and A. Pascucci, "DLR's torque-controlled light weight robot iii - are we reaching the technological limits now?" in *Proc. of the Intl. Conf. on Robotics and Automation*, 2002, pp. 1710–1716.

[9] D. Aarno, "Autonomous path planning and real-time control - a solution to the narrow passage problem for path planners and evaluation of real-time linux derivatives for use in robotic control," Master's thesis, Department of Numerical Analysis and Computer Science (NADA), KTH, Sweden, 2004, TRITA-NA-E04006.

[10] S. Munir and W. J. Book, "Internet-based teleoperation using wave variables with prediction," *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 2, pp. 124–133, Jun 2002.

[11] J. Craig, *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Pub. Co., Reading, 1986.

[12] M. Walker and D. Orin, "Efficient dynamic computer simulation of robotic mechanisms," in *Transactions of the ASME – Journal of Dynamic Systems, Measurement and Control*, vol. 104, 1982, pp. 205–211.

[13] I. Ishii and M. Ishikawa, "Self windowing for high-speed vision," *Systems and Computers in Japan*, vol. 32, no. 10, pp. 51–58, 2001.