2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Using COTS to Construct a High Performance Robot Arm

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

***Abstract*— In this paper we present a design study and technical specifications of a high performance robotic manipulator to be used for ball catching experiments using commercial off-the-shelf (COTS) components. Early evaluation shows that very good performance can be achieved using standardized *PowerCube* actuator modules from Amtec and a standard workstation using CAN bus communication. Implementation issues of low-level control and software platform are also described, as well as early experimental evaluation of the system.**

I. Introduction

Robotic systems fast enough to dynamically manipulate ballistic objects have been around since the first ball-catching and ping-pong playing systems were implemented 15 to 20 years ago (as reported by e.g. [1] [2] [3]). However, such systems have only recently become commercially available and for many research purposes, special experimental setup requirements might still pose constraints that are not easily fulfilled by standard models. Therefore, high performance manipulation setups often require a tedious and costly design process where each implementation is made up of custom solutions.

The present paper intends to show how off-the-shelf components can be deployed to achieve rapid prototyping for very competetive performance. 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. All components used are readily available commercial products. A photo of the final implementation is shown in Fig. 1, and a summary of the specifications and achieved performance is presented in Table X, at the end of this document.

II. Design Procedure

This section provides an initial analysis of the requirements for a system to perform ball-catching. From an analysis of requirements a design for the system is developed. The design is simulated to verify the performance.

A. Experimental Requirements

The main type of experiment that the proposed platform needs to cater for involves 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 accomodate variations in accuracy corresponding to catching the ball within a 60×60 cm window. From these basic requirements it is possible to compute flight time and velocities for the scenario, as summarized below:

Fig. 1. The high-performance manipulator.

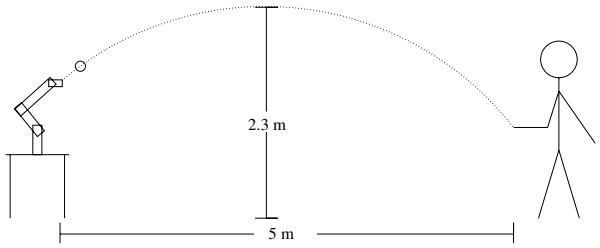


Fig. 2. Schematic of ballcatching experiment.

- 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.
- The ball should be caught if 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 of the desired requirements of the scenario is use of standard video cameras for trajectory estimation. Using 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 an additional trajectory estimation, resulting in a 60 ms estimation time. However, limited accuracy of the cameras will mean that more, probably as many as 10 images might be necessary (c.f. [4]), and the time delay from initiation of a throw to initial trajectory estimate might be 200 ms. This particular setup is also intended to be used for teleoperated catching, so we also have to allow for extra reaction time as we include a human in the loop. This might add an additional 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. Depending on the type of end effector[1], the positioning has to be within one or a few centimeters of error from the ball trajectory. 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 travelled 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 term 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 mechanical 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 duly 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 also 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 has to fulfill can be summarized as:

- 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.

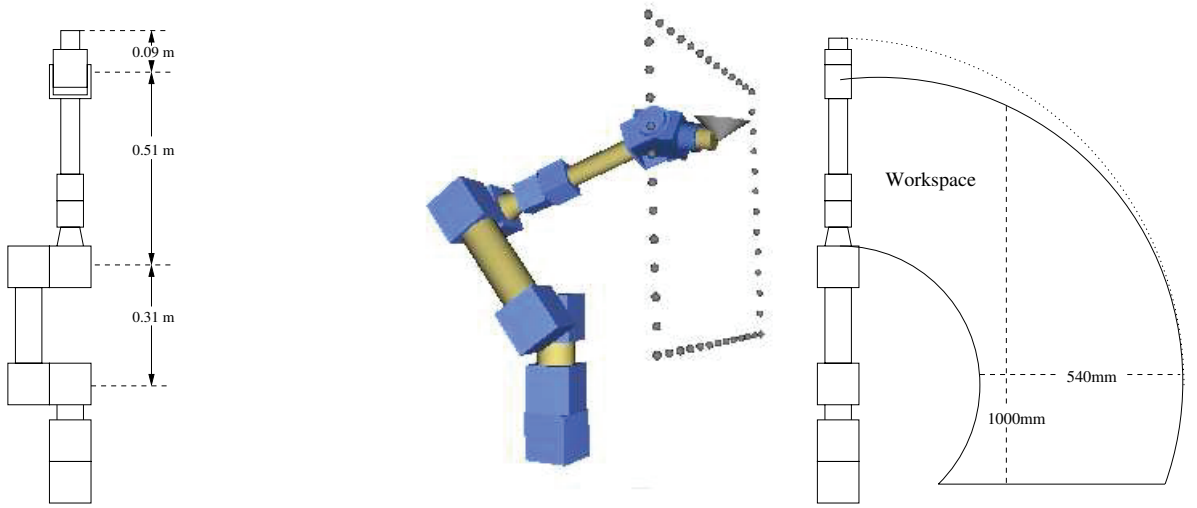
C. Designed Solution

There are a number of fairly fast robotic manipulators available, like for instance the DLR developed Kuka Light Weight Arm [5]. It has been shown to be fast enough to catch thrown balls autonomously [4], 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 be able 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, it was decided to construct a 6 DoF arm, and to examine if this could be done using *PowerCube* modules from the German manufacturer Amtec. These modules are available off the shelf and allow for rapid prototyping. In addition the range of modules clearly include some that have specifications which are adequate for the present application (See Section III-A). In addition, the modules have a built-in controller that can be used for embedded safety and low level control.

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 industrially available). This is not only a configuration that allows for very good dynamic performance (see Section II-D), but as it is a widely used and studied configuration, several implementation issues have already been solved, thus making the design process considerably faster. For example, the closed form solution for inverse kinematics 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.

[1]Since the initial experiments will not be concerned with grasping, a simple passive end effector like a net will be employed.



(a) Dimensions (b) 3D rendering of arm and operational window (c) Workspace with tool oriented towards user.

Fig. 3. The manipulator, constructed with Amtec PowerCubes.

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 accomodate 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.

It was decided that the computer doing the direct control should run RTAI, a real time Linux system that has showed good performance in previous studies [6]. The control computer will perform the trajectory generation and be responsible for dynamic and kinematic calculations. A secondary computer will be used for the user interface. The communication between the two 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 by the real-time low-level controller for best performance. The connection is made over UDP/IP, as this has been shown to give much better control performance than TCP/IP over an internet connection (see e.g Munir and Book [7] for a complete evaluation). The connection to the user interface will not need hard realtime performance, but the smaller the time lag can be made, the better the performance. In early experiments over 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
- Linux, preferably RTAI for control computer
- Communication over several parallel CAN connections.

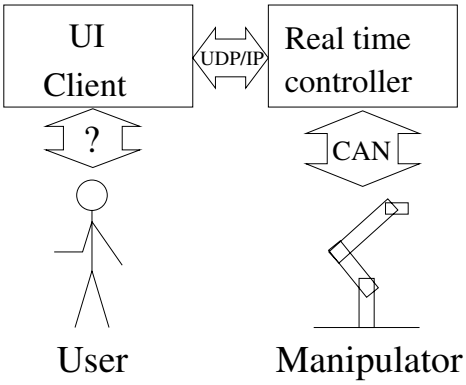


Fig. 4. Schematic of the connection architecture.

TABLE I

SIMULATED PERFORMANCE OF ROBOT ARM

Since performance is highly dependent of the current position of the arm, all values are given as their lower limit within the window. The performance is thus equal to or better than the stated values for all points in the window. Where nominal performance differs greatly from worst-case, nominal performance is presented in braces. The traveltimes were calculated using a simulation including a non-optimized controller algorithm, and are therefore conservative approximations.

Endpoint acceleration	$> 100\ m/s^2$	$(140 m/s^2)$
Endpoint velocity	$> 5\ m/s$	$(7 m/s)$
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	$\pm 1\ mm$	

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.

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
2ond 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)	$\pm$0.00035 rad
2	267 Nm	4.1 rad/s (238^o/s)	$\pm$0.00035 rad
3	134 Nm	8.2 rad/s (470^o/s)	$\pm$0.00035 rad
4	23 Nm	8.2 rad/s (470^o/s)	$\pm$0.00035 rad
5	35 Nm	4.3 rad/s (248^o/s)	$\pm$0.00035 rad
6	8 Nm	6.2 rad/s (356^o/s)	$\pm$0.00035 rad

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 Table II and III.

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

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	0	θ_1
2	$-90\,^\circ$	0	0	θ_2
3	0	0.31 m	0	θ_3
4	$-90\,^\circ$	0	0.51 m	θ_4
5	$-90\,^\circ$	0	0	θ_5
6	$90\,^\circ$	0	0	θ_6

To test the reliability of 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. This implies that the arm assumes the last commanded position. In the second security experiment, illegal positioning commands were purposefully 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 bus communication link.

In order to avoid collisions, both with itself and environment, the angles of the different joints have been limited to the values showed 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.

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 2.6.9 Kernel was patched with RTAI 3.2 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 1 ms, which is sufficient. The average jitter for the control algorithm is 60 μs, which is significantly less than the modules' latency of up to 600 μs.

Inverse kinematics and dynamics are calculated using a C implementation of the analytical solution for a Puma arm from Craig [8], and the forward dynamics are calculated using the second algorithm proposed by Walker and Orin [9]. 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.

Position 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. The maximum acceleration is limited by a preset limit value[2] (presently 16 rad/s^2) and the maximum achievable acceleration, computed by calculating the resulting acceleration with maximum torque and taking away a small safety margin. 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 forward dynamics function that determines the necessary torques to follow the trajectory. These torques are converted to currents and sent to the actuator modules. For a schematic of the control scheme, see Fig 5.

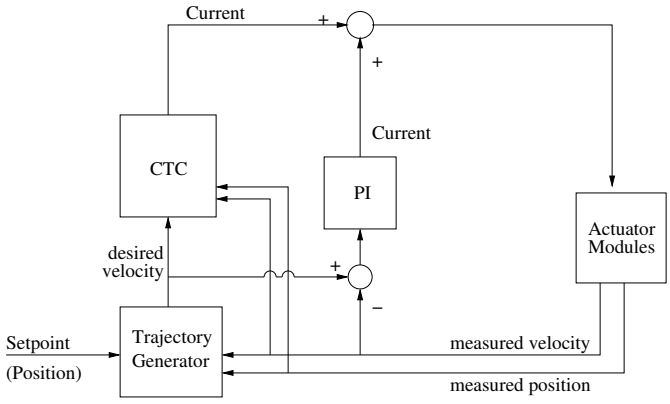


Fig. 5. Schematic of controller

[2]The limits on velocity and accelaration 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.

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 realationship. Empirical trials have shown that good performance is achieved when the gain balance is set so that the PI controller produces approximately 30 % of the control signal.

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 milimeter 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 observers 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 workspace vertically (distance 60 cm) in both directions within 0.39 s, which was predicted in the simulations. As for the other movements, only times down to 0.5 s have been verified, as this is enough for our application and we want to minimize mechanical stress on the equipment. The outermost joints are slightly slower then the inner ones, so the final angular alignment of the end effector takes slightly longer than the positioning in some configurations.

C. Control loop times

Although the PowerCube modules are specified by the manufacturer to be able to handle CAN bus communication up to 1 Mbit/s, experiments showed that this rate can not be maintained continously. Especially when controlling several modules on a single CAN bus, there is a tendency for CPU overload/overheat in the modules. Overloading results in an unrecoverable error that requires a shutdown and cooldown before operation can be resumed. The actual achievable communication speeds are presented in Table VI. These are slightly slower speeds than anticipated from the specifications (see Table VII), but still fast enough for acceptable control. The time to complete a communication loop consists of the 0.134 ms needed to send a CAN message at 1 Mbaud (or 0.268 ms at 500 kbaud), 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 implemetation, a control loop frequency of 600 Hz for the inner 3 cubes and 200 Hz for the outer 3 cubes was used. 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 neglible.

TABLE VI

Empirical control loop frequencies over the CAN bus.

	Modules per CAN controller card	
	1	3
Control frequencies at 1 Mbaud		
with velocity polling	Overload	Overload
without velocity polling	1889 Hz	Overload
Control frequencies at 500 kbaud		
with velocity polling	708 Hz	238 Hz
without velocity polling	1256 Hz	417 Hz

TABLE VII

Theoretical control loop speeds over the CAN bus

	Modules per CAN controller card			
	1	2	3	6
Cycle periods at 1 Mbaud				
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 kbaud				
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 Mbaud				
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 kbaud				
with velocity polling	637 Hz	467 Hz	311 Hz	156 Hz
without velocity polling	1265 Hz	935 Hz	621 Hz	311 Hz

D. 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 parts of the communication, as well as all measurements are time-stamped in order to enable correction for time-lags over the communication link.

V. Ball Catching Experiments

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

A. Experimental Setup

The manipulator was fitted with a simple end effector, consisting of a passively damped cardboard basket with a diameter of 14 cm (see Fig. 7). Into this we threw a soft juggling type ball from a distance of approximately 4 m. To ensure repeatability, the ball was launched using a mechanical launcher with a precision of ± 10 cm at the specified distance (see Fig 6).

Fig. 6. 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 7). The ball tracking was done by using an extended Kalman filter (EKF), as described in [4].

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

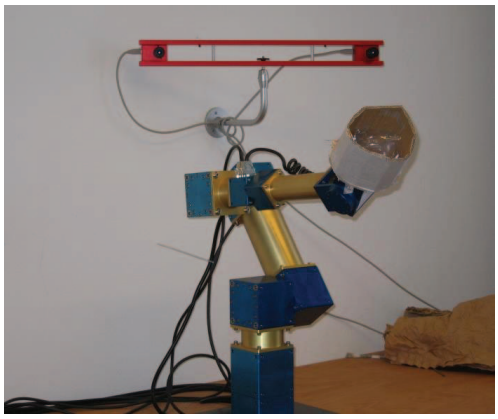


Fig. 7. The manipulator with cameras and ball-catching end effector.

center of mass for the marked pixels was calculated and used as the ball centroid. Since subwindowing schemes have shown to be very efficient to significantly reduce the time needed in segmenting moving objects (c.f. [10]), one was applied to our implementation as well. 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 could 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 a plane which corresponds to 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 32 balls that hit within the operating window, with an average distance of 24 cm from the manipulator's starting position, 25 of 32 balls were caught.

VI. Conclusion

The present paper has 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. Detailed Simulation Results

In simulation, the times needed to complete several different motions were calculated for a 0.5 kg payload (see table VIII). This corresponds to carrying the weight of the power cables connected to the robot. Simulation results show

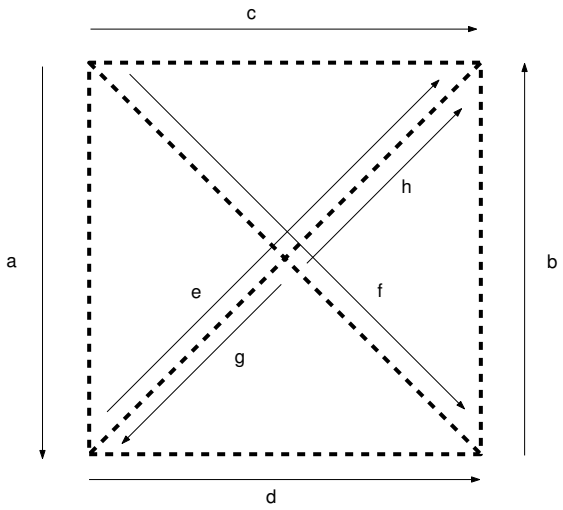


Fig. 8. The distances across the workspace measured for travel times.

TABLE VIII
TRAVEL TIMES

Moving from standstill to standstill with a 0.5 kg payload along different paths in the workspace window. For definitions of distances, see Figure 8.

Distance	Traveltime
a (top-bottom)	0.37 s
b (bottom-top)	0.36 s
c (left-right)	0.23 s
d (left-right)	0.23 s
e (diagonal up)	0.37 s
f (diagonal down)	0.37 s
g (mid-lowerleft)	0.26 s
h (mid-upperright)	0.22 s

that the time constraint of traveling from one position to another in less than 0.5 s can be fulfilled.

B. 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 IX. Please note that these are the actual prices paid, and that there is no guarantee for future availabilty at these same prices.

C. Experimental verification of performance

On the actual platform, due to safety precautions, velocities higher than those necessary have not been verified. motions from the center of the workspace to the edges moving horizontally, vertically and diagonally have been verified at less than 0.5 seconds. Additionally, the same travel times have been verified when traversing the entire workspace from edge to edge. This was measured by inserting a timer into the control loop that measures the time from starting the control loop in a certain position until the control loop reports both that the position is within 5 mm of goal position and velocity is less than $1^o s^{-1}$ for all joints.

TABLE IX
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

TABLE X
OVERALL SPECIFICATIONS FOR THE ARM

Total reach	0.91m
Max end effector velocity	7 m/s
Max end effector acceleration	140 m/s^2
Max power consumption	5 kW
Moving mass[a]	10.6 kg
Control bus	CAN (4 channels)
Control frequency[b]	600/1200 Hz
External connection	UDP/IP
Control space	Joint/Cartesian
Control types	Position/Velocity

(a) Moving mass only indicates mass that makes translational motion, and thus has a large inpact on the inertial load.
(b) This depends on the velocity measuring strategy, see Section IV-C

ACKNOWLEDGMENT

The research presented in this paper is funded in full by the 6th EU Framework Program, FP6-IST-001917, project name Neurobotics.

REFERENCES

[1] R. Andersson, "Dynamic sensing in a ping-pong playing robot," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 6, pp. 728–739, December 1989.

[2] H. Hashimoto, F. Ozaki, K. Asano, and K. Osuka, "Development of a pingpong robot system using 7 degrees of freedom direct drive arm." in *1987 International Conference on Industrial Electronics, Control, and Instrumentation*, Nov 1987, pp. 608–615.

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

[4] 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.

[5] 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.

[6] 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.

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

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

[9] 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.

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