Proceedings of the
Austrian Robotics Workshop ’13

University of Applied Sciences Technikum Wien

Höchstädtplatz 6
A-1200 Wien

Wilfried Kubinger, Alexander Hofmann, Friedrich Praus (Eds.)
ISBN: 978-3-200-03095-4

May 2013, Vienna

1 Preface

The Austrian Robotics Workshop seeks to bring together researchers, students, professionals and practitioners to discuss recent developments and challenges in robotics and its applications. Since the early days of the Austrian RoboCup workshop back in 2006, the workshop has been a regional platform for networking and for the exchange of ideas and expertise between (mainly Austrian) universities, universities of applied sciences, research centers and companies.

The contributions for the 2013 workshop cover a wide range of topics, ranging from industrial robots, mobile and service applications to humanoid robots. A student session is dedicated to ongoing or early work to encourage Master- and PhD-students to present and discuss their research topics.

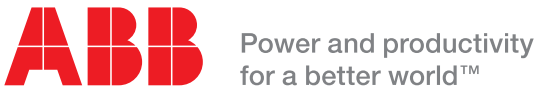
An industrial exhibition has been organized in conjunction with the workshop in order to discuss or initiate collaborations between academia and industry.

We would like to thank all authors, reviewers, presenters and speakers for their contributions to the workshop. Furthermore, we would like to thank UAS Technikum Wien, IEEE Austria and IEEE Robotics and Automation Austria Section for their support and contributions to the workshop.

Finally, we would like to thank ABB AG, Robotics Österreich, Brown Boveri Straße 1, 2351 Wr. Neudorf for their generous support of the workshop dinner at Martin Sepp Heurigen.

Wilfried Kubinger, Alexander Hofmann and Friedrich Praus

Vienna, May 2013



2 Program Committee

Horst Bischof	Graz University of Technology
Michael Hofbaur	UMIT
Jens Knoop	Vienna University of Technology, Dept. of Computer Science
Wilfried Kubinger	University of Applied Sciences Technikum Wien
Michael Naderhirn	AeroSpy Sense & Avoid Technology GmbH
Pavel Petrovic	Comenius University, Department of Applied Informatics
Andreas Pichler	PROFACTOR GmbH
Friedrich Praus	University of Applied Sciences Technikum Wien
Peter Rössler	IEEE Austria Section
Fritz Schmöllebeck	University of Applied Sciences Technikum Wien
Markus Schordan	University of Applied Sciences Technikum Wien
Lukas Silberbauer	taurob GmbH
Gerald Steinbauer	Graz University of Technology
Markus Vincze	Vienna University of Technology

3 Additional Reviewers

Angerer, Arthur
Capco, Jose
Gruber, Christoph
Ikeda, Markus
Kandlhofer, Martin
Maurer, Johannes
Mühlbacher, Clemens
Neumayr, Richard

4 Invited Talks

Branislav Borovac	University of Novi Sad
Daniele Nardi	Sapienza Università di Roma

5 Industry Talks

Martin Kohlmaier / ABB AG	Trends and Innovations from ABB Robotics
Ronald Naderer / FerRobotics Compliant Robot Technology GmbH	Robot Handcraft (R) - Robotisation of manual work
Martin Schenk / taurob GmbH	Overturn Prevention of a Mobile Robot with a Multi-DoF Arm

6 Table of Contents

7 Author Index

Energy Efficiency and smoothness in robotics trajectory planning: numerical simulation and comparison

Paolo Boscariol[3], Andrea Gasparella[1], Alessandro Gasparetto[3], Nicola Lever[1], Dario Richiedei[2], Alberto Trevisani[2] and Renato Vidoni[1]

***Abstract*— In this paper, the most widely adopted industrial off-line non model-based trajectories together with optimum time-jerk and time-energy algorithms are considered and evaluated in terms of energy efficiency and smoothness.**

First of all, a robotic dynamic simulator able to run different laws of motion, to simulate the robot dynamic behavior and to evaluate the amount of mechanical energy, torque and jerk, has been developed and implemented in Matlab.

After that, both point-to-point and pick-and-place trajectories have been simulated by comparing different motion laws whose results have been evaluated and ranked from both the energy efficiency and smoothness point of view.

Finally, a performance index able to take into account the energetic and vibrational performance has been defined to compare the different trajectory planning algorithms.

I. INTRODUCTION

Trajectory planning is a fundamental issue for robotics and mechatronics applications. Indeed, the ability to generate trajectories with prescribed features is a crux to ensure effective results in terms of quality and feasibility of performing the required motion. Different criteria aimed at optimizing the motion have been proposed in literature [1],[2] and attention has been mostly paid on performing fast motions and, eventually, ensuring adequate smoothness. In contrast, only few works address the optimization of the energy consumption, although energy-based optimal trajectory planning criteria can cover an important role in the frame of a green-mechatronic approach and sustainable vision.

Indeed, the concept of energy efficiency and conservation in automation industry and robotics has become in focus only in the recent years, due to the increasing of the energy costs and to the problems and rules fixed to limit or control the climate change. Thus, at today, the energy saving target is not just a mere economic implication, but also an ethical issue and a possible add-on for the market competitiveness of the industrial products and applications. Among the techniques to reduce energy consumption in robotic and mechatronics systems, the development of energy efficient trajectories shows promising results since it does not rely on hardware modifications and therefore can be easily implemented in both new and existing systems to improve their efficiency.

Besides energy efficiency optimization, for planning an effective trajectory other features have to be achieved. In particular, it has to be taken into account that severe vibrations can arise in manipulators when they are moved along a non-smooth trajectory. In that case worsening of accuracy, premature joint wear and mechanical failures might occur [3].

To test this purposes, in this work, the simultaneous evaluation of both the energy efficiency and the smoothness in off-line trajectory planning in robotics and, more in general, in industry is addressed.

In literature, extensive surveys on trajectory planning techniques can be found [1], [2], but rarely a comparative performance analysis and a performance index definition have been proposed (e.g. [4]). In general, a possible solution to accomplish a given task using a robotic manipulator is to synthesize the optimal motion with respect to a relevant criterion. Thus, focusing on generating off-line movements to perform tasks known a priori and in a defined environment, it is possible to find different optimality criteria based on the minimization of the execution time, actuator effort or jerk.

A fundamental distinction between the methods available in literature is the use of a model-based or of a model-free approach. Model-based approaches can achieve good results (e.g. [5], [6]) in specific cases but they lack of generality, which is a fundamental requirement for most industrial applications. As a matter of fact, usually, most industrial facilities are not adequately modeled to address model-based approaches, and the personnel training investment is not reputed to be profitable. Therefore model-free approaches, as the ones considered in this paper, are more appealing for todays market. Thus, the most significant off-line non-model based methods and algorithms currently adopted in industrial robotics and mechatronic systems are here considered. In particular, both the state-of-the-art trajectory planning algorithms such as trapezoidal and double-s methods [7], and ad-hoc developed methods with high orders of continuity or synthesized through optimization functions, are considered.

The investigated trajectory planning techniques are evaluated, compared and ranked both in terms of energy costs for clearly quantifying the possible performance enhancement and energy savings, and smoothness to evaluate the capability to provide fast motion while reducing low induced vibrations. Finally, a performance index synthesizing both energy efficiency and smoothness is proposed to provide a

*This work is supported by the Free University of Bolzano under the project TN5050 - Energy Efficiency Techniques for Mechatronic and Robotic Systems

[1] R. Vidoni, A. Gasparella and N. Lever are with the Faculty of Science and Technology, Free University of Bozen-Bolzano, Bolzano, Italy `corr. author: renato.vidoni, at unibz.it`

[2] D. Richiedei and A. Trevisani are with the DTG of Vicenza (I), University of Padua (I)

[3] P. Boscariol and A. Gasparetto are with the DIEGM of the University of Udine (I)

straightforward comparison of all the motion laws investigated.

II. DYNAMIC SIMULATOR

In order to perform the comparison of the different motion laws, an ad-hoc dynamic simulator for robotic systems has been implemented and developed in Matlab.

The dynamic simulator takes into account the geometrical and inertial parameters of the robot in use; in particular, a Newton-Euler approach has been implemented.

The simulator allows both selecting among classical rigid-link robots, e.g. Anthropomorphic and cartesian, or creating particular robots starting from a single-link configuration. Classical and ad-hoc trajectory primitives can be run and the system output are the position, velocity, acceleration and jerk profiles at the joints, and the actuator and system effort in terms of requested torques, work and power.

Thanks to this simulator, a straightforward evaluation of the required effort for each trajectory can be performed.

In Fig.1, the simulator user interface and an example of the simulator result experimental validation, i.e. comparison of the simulated and measured torque on a joint of a real robot, are shown.

Two robots have been chosen for the numerical evaluation: a cartesian robot and an anthropomorphic robot.

Tab. I, II and III report the main DH and mechanical parameters of the robots.

TABLE I

DH TABLE, CARTESIAN AND ANTHROPOMORPHIC ROBOTS

Cartesian	$\alpha[deg]$	$a[m]$	$\theta[deg]$	$d[m]$
Base frame	0	0	0	0.75
Joint1	90	0	0	d_1
Joint2	90	0	90	d_2
Joint3	0	0	0	d_3
Joint4	-90	0	θ_4	0
Joint5	90	0	θ_5	0
Joint6	0	0	θ_6	0.1
Anthrop.	$\alpha[deg]$	$a[m]$	$\theta[deg]$	$d[m]$
Base frame	0	0	0	0.75
Joint1	90	0	θ_1	0
Joint2	0	0.71	θ_2	0
Joint3	90	0	θ_3	0
Joint4	-90	0	θ_4	0.859
Joint5	90	0	θ_5	0
Joint6	0	0	θ_6	0.1

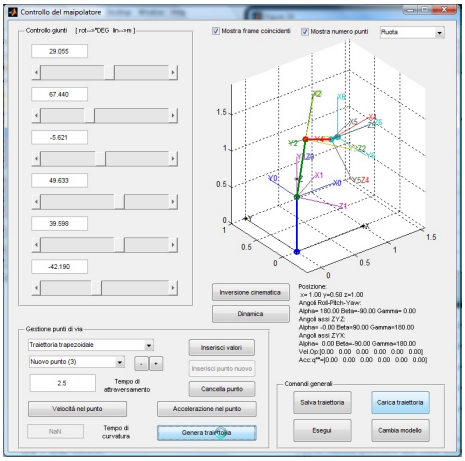
TABLE II

ROBOT ARMS MASSES

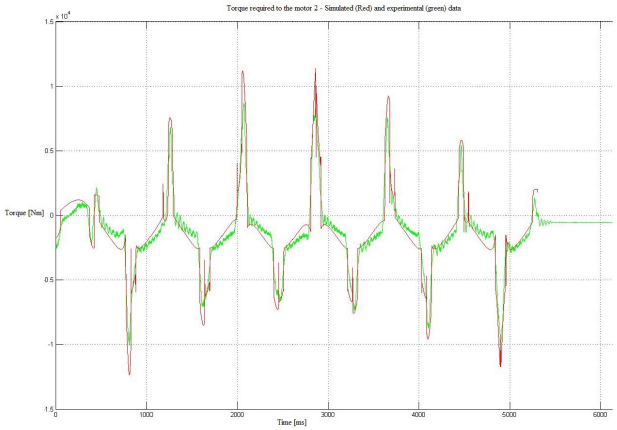
Arm	*base*	1	2	3	4	5	6
Cartesian [kg]	10	10	10	10	10	10	10
Anthrop. [kg]	190	45	45	40	18	8	4

III. PRIMITIVE TRAJECTORIES

Different industrial relevant robots and paths have been evaluated in order to effectively compare the motion laws.



(a) Dynamic simulator



(b) Simulator validation

Fig. 1. Dynamic simulator: a- user interface; b- validation

As far as robot type is concerned, three main systems have been simulated: a single-link single-motor system, an Anthropomorphic robot and a Cartesian robot. As for the paths both point-to point and complex path motions have been considered.

As regards as the *industrial trajectory* primitives, the following and most exploited methods have been evaluated [7]:

- Trapezoidal (T), linear trajectory with parabolic blends (three segments).
- Double-S (2S), trajectory with double S velocity profile (seven segments).
- Harmonic (H), trigonometric trajectory with an acceleration profile proportional to the position profile with opposite sign.
- Cycloidal (C), trigonometric trajectory with a continuous acceleration profile.
- Quadratic poly (PQ), parabolic trajectory.
- Cubic poly (P3), polynomial trajectory of degree three, four parameters.
- Quintic poly (P5), polynomial trajectory of degree five,

TABLE III

INERTIA MATRIX AND CENTER OF GRAVITY (COG) FOR EACH ARM OF THE ANTHROPOMORPHIC ROBOT

Arm	$base$	1	2	3
Inertia matrix $[kg \cdot m^2]$	$\begin{pmatrix} 70.59 & 0 & 0 \\ 0 & 70.59 & 0 \\ 0 & 0 & 18.12 \end{pmatrix}$	$\begin{pmatrix} 2.60 & 0 & 0 \\ 0 & 1.71 & 2.67 \\ 0 & 2.67 & 1.98 \end{pmatrix}$	$\begin{pmatrix} 0.84 & 0 & -9.09 \\ 0 & 10.56 & 0 \\ -9.09 & 0 & 9.91 \end{pmatrix}$	$\begin{pmatrix} 1.75 & 0 & 0.51 \\ 0 & 2.05 & -1.04 \\ 0.51 & -1.04 & 1.19 \end{pmatrix}$
CoG [m]	$\begin{pmatrix} 0 & 0 & -0.55 \end{pmatrix}$	$\begin{pmatrix} 0 & -0.125 & 0 \end{pmatrix}$	$\begin{pmatrix} -0.36 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0.08 & 0 & 0.09 \end{pmatrix}$
Arm	4	5	6	
Inertia matrix $[kg \cdot m^2]$	$\begin{pmatrix} 0.66 & 0 & 0 \\ 0 & 0.03 & 0 \\ 0 & 0 & 0.65 \end{pmatrix}$	$\begin{pmatrix} 0.016 & 0 & 0 \\ 0 & 0.012 & 9.95 \\ 0 & 9.95 & 0.007 \end{pmatrix}$	$\begin{pmatrix} 0.004 & 0 & 0 \\ 0 & 0.004 & 0 \\ 0 & 0 & 0.0006 \end{pmatrix}$	
CoG [m]	$\begin{pmatrix} 0 & 0.16 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & -0.003 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & -0.027 \end{pmatrix}$	

six parameters.

- SPLINE, spline trajectory with cubic primitive.

Among the *optimum trajectory* planning techniques, the following trajectory primitives have been implemented and simulated:

- Minimum acceleration (effort), trajectory that minimizes the integral of the square value of the joint accelerations,
- Continuous-jerk 445 [8],
- Minimum time-jerk minS3 [1], [4], this algorithm is based on cubic splines.
- Minimum time-jerk minBS5 [1], [4], this algorithm is based on quintic splines

In the Minimum time-jerk trajectories, either the parameters are properly chosen or the trajectory time is scaled or elongated through a time scale process to obtain the desired motion time. Moreover, these two optimum time-jerk trajectories, minS3 and minBS5, based on the minimization of a two-term objective function, have been simulated to evaluate their effectiveness in terms of energy and energy-jerk efficiency. The jerk contribute is taken into account in the minimization function as the integral of its squared value.

As previously stated, only off-line non-model based methods have been considered since their synthesis is independent from the particular robotic system under investigation and does not rely on the knowledge of any dynamic parameter or model of the system.

IV. COMPARISON

The first test has been made by considering a point to point motion along the X-axis of a cartesian robot, i.e. a single-link, single-motor system; the total displacement was 0.5 m. The gravity acceleration, 9.81 m/s^2, has been simulated on the Z-axis while friction has been considered in its static and dynamic effect. Tab. IV shows the numerical results.

The $T_{1/2}$ and $T_{1/3}$ trajectories are symmetric trapezoidal trajectories with the λ parameter, which defines the acceleration time, set to 1/2 and 1/3 of the motion time respectively.

The comparison of the results shows that, for a point-to-point linear motion, the P3 trajectory is the most efficient in terms of energy, measured through W. This can be viewed as a confirmation of the properties of this polynomial primitive

TABLE IV

POINT TO POINT MOTION WITH ZERO INITIAL AND FINAL VELOCITIES AND ACCELERATIONS FOR A SINGLE-LINK SINGLE-MOTOR SYSTEM - MOTION TIME = 6 S; P = POWER, W = WORK, τ = TORQUE, J = JERK

	$T_{1/2}$	$T_{1/3}$	2S	H	C	P3	P5
P_{RMS}	57.45	39.39	42.89	34.08	61.78	**30.89**	52.74
W	11.51	9.87	10.15	9.45	13.09	**8.98**	12.04
τ_{RMS}	100.00	92.09	102.21	87.66	110.52	**87.46**	102.99
$\int J^2$	∞	∞	411.08	∞	195.01	∞	**180.72**
J_{RMS}	∞	∞	20.27	∞	13.96	∞	**13.44**

that minimizes the quality index $\int_0^{t_f} \tau^2 dt$, where t_f is the total time of the trajectory.

Since if the jerk is limited or minimized the tracking accuracy increases and the excitation of the resonant frequencies is reduced, this value has an important significance in the trajectory algorithm performance.

As far as the jerk is concerned, only three trajectories show a finite value. Among the motion laws with finite jerk, the 2S trajectory is the less energy expensive, providing an increase of the 13% compared with the energy required by the P3.

The second test has been made by performing a pick and place motion with both a Cartesian and an Anthropomorphic robot. The same motion time has been considered for all the motion laws. The points to follow in the operative space are reported in Tab. V.

TABLE V

POINTS OF THE PICK AND PLACE MOTION IN THE CARTESIAN SPACE

Point	X [m]	Y [m]	Z [m]
P0	0.5	0.5	0.5
P1	0.5	0.5	0.25
P2	0.5	0.5	0.5
P3	0.2	0.1	0.5
P4	0.2	0.1	0.25
P5	0.2	0.1	0.5
P6	0.5	0.5	0.5

Tab. VI shows the corresponding joint values, solution of the inverse kinematics for the Anthropomorphic robot.

As can be seen, the angular values of two of the wrist joints remain constant along the whole trajectory.

TABLE VI

POINTS OF THE PICK AND PLACE MOTION IN THE JOINT SPACE FOR THE ANTHROPOMORPHIC ROBOT

Point	J1 [deg]	J2 [deg]	J3 [deg]	J4 [deg]	J5 [deg]	J6 [deg]
P0	45	61.69	-36.15	180	25.54	225
P1	45	38.82	-28.50	180	10.32	225
P2	45	61.69	-36.15	180	25.54	225
P3	26.57	81.09	-73.49	180	7.61	225
P4	26.57	31.30	-57.78	180	-26.48	225
P5	26.57	81.09	-73.49	180	7.61	225
P6	45	61.69	-36.15	180	25.54	225

Tab. VII and Tab. VIII report the simulation results in terms of energy and smoothness parameters.

Even if the performed motion is a pick and place, the movements made by the joints of the two robots are substantially different. Indeed, in the case of a cartesian robot, the main joint movements are point-to-point thus no continuous motion along the linear joints is planned. Indeed, if the motion along the Z-axis is considered, only the vertical linear joint is in charge to perform the action. On the contrary, in the Anthropomorphic robot, some joints have to move along the whole trajectory.

Thanks to this consideration, the results in Tab. VII and Tab. VIII can be better understood.

For the Cartesian robot movement (see Tab. VII), the P3 trajectory allows again the best result in terms of required power, work and torque. Among the optimal trajectories, it can be appreciated how the minimum time-jerk trajectories allow the best performances both in work and in jerk content.

If smoothness is also accounted for, the 2S represents, among the other industrial trajectories, the best compromise since it allows a good behavior in terms of energetic performances and a finite jerk RMS value.

As for the Anthropomorphic robot, the analysis of Tab. VIII leads to different considerations: the minimum time-jerk trajectories allow the best performance both for the energy and jerk parameters.

In both the simulations, the worst "energy" case is represented by the 445 law. This result can be easily justified since the overall path length to be run by the robot has to be noticeably increased to allow the smoothness required by the trajectory algorithm.

In Tab. IX the laws of motion are ranked and the deviation with respect to the best one is given in percentage for the Anthropomorphic robot.

As can be appreciated from the results, important savings and performance enhancements can be achieved by implementing the proper law of motion. Indeed, even if the torque requirements does not show important deviation from the minimum value, i.e. the minBS5, both the work, W, and jerk, J_{RMS}, values show great differences. As an example, the minS3 trajectory results the best choice in terms of energy and allows, for the simulated path, a reduction of more than the 10% with respect to a classical and widely adopted SPLINE, while providing a jerk finite value.

TABLE IX

TRAJECTORY RANK

Position	W		τ_{RMS}		J_{RMS}	
	Traj	Inv	Traj	Inv	Traj	Inv
1	minS3	-	minBS5	-	minBS5	-
2	P3	0.3%	P3	2.1%	minS3	14%
3	H	1.0%	mins3acc	2.2%	SPLINE	39%
4	T1/3	1.4%	H	2.3%	445	59%
5	2S	2.2%	T1/3	2.7%	P5	220%
6	T1/2	3.8%	minS3	3.1%	C	233%
7	P5	4.1%	SPLINE	3.1%	2S	384%
8	C	5.1%	2S	3.4%	minS3acc	1455%
9	minBS5	9.8%	T0.5	3.4%	-	-
10	minS3acc	10.2%	P5	4.2%	-	-
11	SPLINE	10.5%	445	4.8%	-	-
12	445	43.4%	C	5.0%	-	-

It can be added that, as a general remark, the optimum methods allow the best results both in terms of energy and jerk, and should be preferred when no point-to-point motions have to be performed. On the contrary, when point-to-point movements are requested along the trajectory, the effect of the optimization is reduced and the "classical" trajectory algorithms that allow a finite jerk value show a good compromise in terms of algorithm complexity and performances.

A. PERFORMANCE INDEX

In order to define and propose a synthetic criteria to classify the performance of a trajectory by taking into account both the energetic and vibrational requirements, a performance index (PI) has been defined.

The 2S trajectory has been chosen as the reference law due to its main characteristics: simplicity, industrial implementation and continuity in acceleration, hence finite jerk value.

The chosen PI takes into account the weighted relative values achieved by the considered trajectory in terms of energy and jerk with respect to the reference ones:

$$PI = k_e * \frac{W_i}{W_{ref}} + k_j * \frac{J_i}{J_{ref}}$$

where k_e+k_j=1. By setting to zero one of the two weights, the motion laws are classified either with respect to the energy efficiency or to the minimum jerk.

In order to be able to compare all the simulated trajectories, thus have a finite PI also for the laws with discontinuous acceleration, infinite jerk values have been included in PI by replacing them in the J_{RMS} with a high but finite upper-limit jerk value. This means that for each infinite peak a finite value has been accounted for a prescribed duration, i.e. $500m/s^3$ for 5 ms.

In this way, all the motion laws can be evaluated and compared on a same benchmark path, e.g. a pick and place or a smooth circular paths, allowing a direct comparison in terms of energy efficiency, smoothness or their combination with respect to the 2S standard law.

TABLE VII

CARTESIAN ROBOT - PICK AND PLACE; MOTION TIME = 6 S

	$T_{1/2}$	$T_{1/3}$	2S	H	C	P3	P5	SPLINE	445	minS3	minBS5
P_{RMS}	141.34	135.24	139.17	134.35	149.96	132.35	145.68	143.96	222.21	145.19	165.66
W	70.83	70.24	70.93	70.29	72.92	69.95	71.86	75.29	97.53	64.95	77.35
τ_{1RMS}	590.62	589.31	590.63	590.12	591.15	590.09	590.84	588.69	591.96	588.43	589.41
τ_{2RMS}	47.06	43.47	48.20	41.41	51.80	40.69	48.43	36.23	45.93	23.78	25.92
τ_{3RMS}	28.28	26.13	28.96	24.90	31.10	24.47	29.09	21.91	27.65	14.33	15.62
τ_{totRMS}	665.96	658.91	667.79	656.43	674.05	655.25	668.36	646.83	665.54	626.54	630.95
$\int J^2$	∞	∞	1231.96	∞	584.55	∞	540.23	114.35	204.62	70.19	50.40
J_{RMS}	∞	∞	14.33	∞	9.87	∞	9.48	4.36	5.84	3.42	2.89

TABLE VIII

ANTHROPOMORPHIC ROBOT - PICK AND PLACE; MOTION TIME = 6 S

	$T_{1/2}$	$T_{1/3}$	2S	H	C	P3	P5	SPLINE	445	minS3	minBS5	minS3acc
P_{RMS}	278.87	262.17	270.37	259.32	297.53	254.68	286.41	268.10	311.93	240.57	260.44	222.88
W	128.60	125.65	126.69	125.12	130.30	124.33	129.02	136.99	177.73	123.93	136.02	136.62
τ_{1RMS}	13.79	13.29	15.19	12.75	15.69	12.78	14.77	10.85	14.22	9.87	10.18	10.05
τ_{2RMS}	434.16	431.57	434.83	430.39	439.10	429.67	436.50	428.79	435.35	421.72	414.73	415.68
τ_{3RMS}	98.88	98.02	99.03	97.67	100.21	97.50	99.44	105.64	104.25	113.50	103.72	114.48
τ_{5RMS}	0.33	0.56	0.59	0.54	0.64	0.53	0.62	0.55	0.73	0.55	0.57	0.54
τ_{totRMS}	547.16	543.44	546.96	541.36	555.65	540.51	551.33	545.84	554.55	525.64	529.2	540.77
$\int J^2$	∞	∞	$2.34\text{x}10^7$	∞	$1.11\text{x}10^7$	∞	$1.02\text{x}10^7$	$1.93\text{x}10^6$	$2.52\text{x}10^6$	$1.29\text{x}10^6$	$9.99\text{x}10^5$	$2.42\text{x}10^8$
J_{RMS}	∞	∞	1976.12	∞	1361.15	∞	1308.48	567.72	648.12	465.31	408.37	6352.12

TABLE X

CARTESIAN ROBOT - PICK AND PLACE - PI

		$T_{1/2}$	$T_{1/3}$	2S	H	C	P3	P5	SPLINE	445	minS3	minBS5
W		70.8	70.2	70.9	70.3	72.9	70.0	71.9	75.3	97.5	64.9	77.3
J_{RMS}		76	76	14.3	50.1	9.9	58.0	9.5	4.4	5.8	3.4	2.9
W_i/W_r		1.00	0.99	1.00	0.99	1.03	0.99	1.01	1.06	1.38	0.92	1.09
J_i/J_r		5.30	5.30	1.00	3.50	0.69	4.05	0.66	0.30	0.41	0.24	0.20
n_{peak}		28	28	0	12	0	12	0	0	0	0	0
w_e	w_j											
0.2	0.8	4.44	4.44	1.00	3.00	0.76	3.44	0.73	0.46	0.60	**0.37**	**0.38**
0.5	0.5	3.15	3.15	1.00	2.24	0.86	2.52	0.84	0.68	0.89	**0.58**	**0.65**
0.9	0.1	1.43	1.42	1.00	1.24	0.99	1.29	**0.98**	0.99	1.28	**0.85**	1.00

TABLE XI

ANTHROPOMORPHIC ROBOT - PICK AND PLACE - PI

		$T_{1/2}$	$T_{1/3}$	2S	H	C	P3	P5	SPLINE	445	minS3	minBS5	minS3acc
W		128.6	125.65	126.69	125.12	130.3	124.33	129.02	136.99	177.73	123.93	136.02	136.62
J_{RMS}		10309	10309	1976	6251	1361	6271	1308	568	648	465	408	6352
W_i/W_r		1.02	0.99	1.00	0.99	1.03	0.98	1.02	1.08	1.40	0.98	1.07	1.08
J_i/J_r		5.22	5.22	1.00	3.16	0.69	3.17	0.66	0.29	0.33	0.24	0.21	3.21
n_{peak}		28	28	0	12	0	12	0	0	0	0	0	0
w_e	w_j												
0.2	0.8	4.38	4.37	1.00	2.73	0.76	2.73	0.73	0.45	0.54	**0.38**	**0.38**	**2.79**
0.5	0.5	3.12	3.10	1.00	2.08	0.86	2.08	0.84	0.68	0.87	**0.61**	**0.64**	**2.15**
0.9	0.1	1.44	1.41	1.00	1.21	0.99	1.20	**0.98**	1.00	1.30	**0.90**	0.99	**1.29**

Tab. X and Tab. XI show the results for different sets of weights for the pick and place trajectory for the Cartesian and Anthropomorphic robots.

The best PI are highlighted in the two tables.

Thanks to the defined PI it is possible to have a direct comprehension of the effectiveness of the chosen trajectory and its possible benefits in terms of performance, if any. If the case with weights equal to 0.5 is considered, the minS3 and minBS5 have the smallest PI while the most implemented industrial trajectories show a very high PI.

V. CONCLUSIONS

In this work the most adopted industrial trajectory planning techniques together with some minimum acceleration and jerk approaches have been considered and compared from an energy-smoothness performance point of view. Different trajectories have been simulated by means of an ad-hoc dynamic simulator and, after that, compared and ranked.

The results show that important savings in terms of energy can be achieved if the proper law of motion is selected, in particular if the minimization of the jerk content is considered as a performance improvement factor.

An energy-jerk performance index has been also defined in order to directly compare the different trajectory algorithms with respect to a classical double-S assumed as the reference.

Future work will cover the evaluation of the possible energy savings with respect to the kind of robot together with the quantification of the possible energy savings by regenerative braking systems.

REFERENCES

[1] A. Gaspaertto, P. Boscariol, A. Lanzutti, R. Vidoni, Trajectory Planning in Robotics, Mathematics in Computer Science, 2012, DOI 10.1007/s11786-012-0123-8

[2] AA Ata, OPTIMAL TRAJECTORY PLANNING OF MANIPULATORS: A REVIEW, Journal of Engineering Science and Technology, 2 (1), 32-54, 2007

[3] P.J. Barre, R. Bearee, P. Borne, E. Dumetz, Influence of a jerk controlled movement law on the vibratory behaviour of high-dynamics systems, Journal of Intelligent and Robotic Systems, 42(3):27593, 2005.

[4] A. Gasparetto, A. Lanzutti, R. Vidoni, V. Zanotto, Experimental validation and comparative analysis of optimal time-jerk algorithms for trajectory planning, Robotics and Computer-Integrated Manufacturing, 28, 164181, 2012

[5] Z. Shiller, Time-energy optimal control of articulated systems with geometric path constraints, J. Dyn. Syst. Meas. Control 11(8), 139143, 1996

[6] B.J. Martin, J.E. Bobrow, Minimum effort motions for open chain manipulators with task-dependent end-effector constraints. Int. J. Robot. Res. 18(2), 213224, 1999.

[7] L. Biagiotti and C. Melchiorri, Trajectory Planning for Automatic Machines and Robots, Springer-Verlag Berlin Heidelberg, 2010.

[8] K Petrinec, Z. Kovacic, Trajectory planning algorithm based on the continuity of jerk, In Control and Automation, 2007. MED07. Mediterranean Conference on, IEEE, pp. 15, 2007

Workspace Analysis of Cooperating Large Scale Manipulators

Johannes Karl Eberharter[1] and Gerhard Kaufmann[1]

***Abstract*— Cooperating large scale manipulators are becoming more and more important to handle long or heavy parts. Therefore, it is important to know the exact workspace to plan a complete heavy lift. The focus of this paper is the visualization of the workspace under the compliance of large scale manipulators and lift boundary conditions. The analysis focuses on the projected two-dimensional workspace where the calculations are performed analytically. The maximum workspace is described with boundary points of the workspace, calculated from intersections of geometric relations, depending on several parameters: dimensions of the manipulators, weight, size and orientation of the load/cross-beam. The workspace boundary is then defined by these boundary points connected via arcs and line segments which visualize the workspace. As an interesting result we can report a separation of the workspace under certain circumstances.**

I. INTRODUCTION

The demand to lift heavy or extreme long parts increased dramatically over the last years. These lifts need large manipulators like harbor mobile cranes, mobile cranes, crawler cranes, etc.. The costs of these machines increase exponentially in size. A common way to overcome these challenges is to utilize two or more large scale manipulators, see Fig. 1. In addition, for mobile manipulators like cranes,

Fig. 1. A multi-crawler crane lift.

the maximum load needs to be reduced by 25 % [1]. However, this can be avoided if the synchronization of the manipulators can be guaranteed. Special automation systems have been developed within the last years to control such cooperating large scale manipulators [2], [3], [4]. It is very common to plan each lift in advance [5], [6]. Therefore the exact workspace of the cooperating manipulators needs to be known. The lift-planning process, in general, had been investigated by Varghese et al. [7]. They defined a visualization of an environmental walkthru, where also the first steps for the best position of the crane was developed. However, a special insight into the workspace analysis was not presented. Several lift planning software packages for cranes [8], [9], [10] are available on the market, but none of them considers the workspace of multi-crane lifts. The novel idea is to look at the workspace of two cooperating large scale manipulators. The workspace of cooperating manipulators is not as simple as known from a conventional industrial robot, as shown below the shape is quite complex. The shape depends on (i) the dimensions and topology of the large scale manipulators, (ii) the weight and (iii) the size of the load. Also new to conventional workspaces of robots, is that the shape will change, depending on (iv) the desired cooperated motion (orientation) of the load. (v) Also, the workspace can change its topology, in other words it can consist of one or two regions. The visualization of the work environment is depicted in different representations in order to provide a better insight of the possible workspace [11]. A brief remark about the function of such machines:

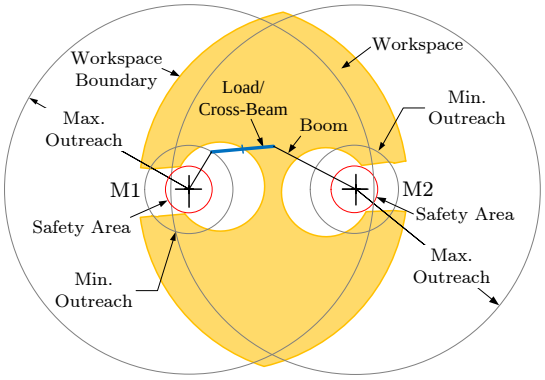


Fig. 2. Definition of the workspace.

Cooperating large scale manipulators do not have six d.o.f. to move a load/cross-beam. The motion is limited to translate in three-space and orientate about a vertical axis. For example, this axis can either be about the hook of a manipulator or about the center of the connecting horizontal line between the hooks [3].

II. BOUNDARY POINTS OF THE WORKSPACE

The workspace will be computed under observance of the given input parameters. Furthermore, the procedure for

*This work was supported by Liebherr-Werk Nenzing GmbH

[1]University of Applied Sciences Vorarlberg, Department of Mechatronics Dornbirn, Austria. Hannes at Eberharter.us Gerhard.Kaufmann at students.fhv.at

calculating the boundary points of the workspace will be discussed in this section. The computation algorithm shall work for offline and online computation; for instance, a path planning or visualization tool. The workspace boundary describes the maximum working range depending on the selected point on the load/cross-beam. Figure 2 shows a possible workspace boundary for two manipulators M1 and M2. The workspace boundary is defined by arc and line

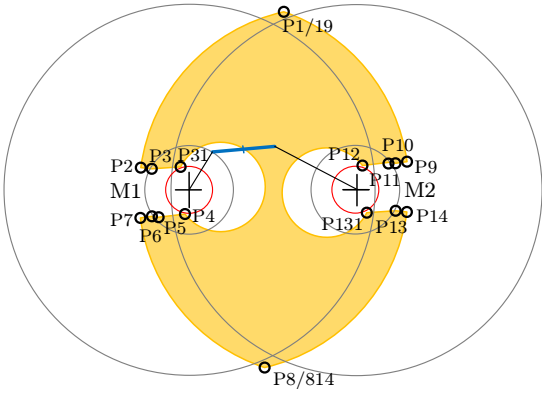


Fig. 3. Boundary points.

segments. In Fig. 3 the points for M1 and M2 are shown. The points $P1-P8$ describe the workspace boundary for M1 and the points $P19-P814$ for M2. All calculations of the points are carried out analytically. To obtain the intersection points of the defined points; lines and/or arcs are used, see Fig. 4. Significant are the load/cross-beam length L_t, the point of interest (POI), which is an arbitrarily elected point, e.g. the center of gravity x_s and the load/cross-beam with its angle φ. To calculate the point $P1$, a line goes through the origin of M1 with the angle of the load/cross-beam. The same applies to the second manipulator M2. The center of the circle (max. working radius of M1) is shifted by the distance x_s in positive direction on the line. For M2, the center of the circle is (within the radius of M2) shifted by the distance $L_t - x_s$ in negative direction on the line. The resulting intersections of the two circles describe the points $P1, P8$ and $P19, P814$. The points $P2-P7$ of M1 and

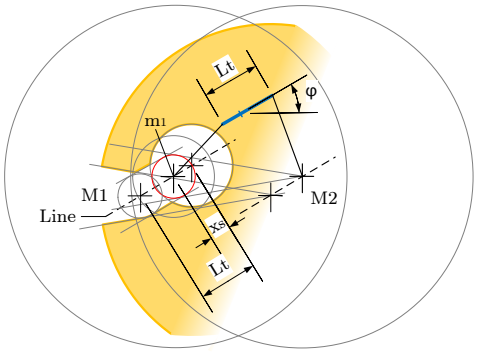


Fig. 4. Auxiliary line calculation, Parallel shift.

$P10-P14$ of M2 (Fig. 3) can be calculated without data of each other. If the maximum outreach of the manipulator changes, then the element of a circle will move as well. In Fig. 5 the element of a circle is shifted to the right (distance between the manipulator is raised). Therefore it must be checked with which line element or element of a circle it intersects. If both elements of a circle cut themselves, one receives the points $P31$ and $P4$. The points in the bracket are congruent and are equated with the intersections. If the points do not exist, this does not mean that they lie beyond the workspace. According to the intersection of the element of a circle several points can lie on top of each other. For the

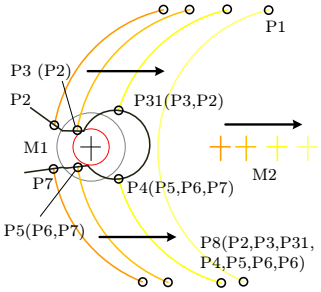


Fig. 5. Change of the manipulator distance.

calculation of the intersections the circles must be shifted by a defined distance. The movement always occurs on a straight line. This line passes through the origin of the respective manipulator and is in parallel to the load/cross-beam (Fig. 4). According to the intersection the center m_1 of the circle is shifted by x_s or $L_t - x_s$ along the line.

III. BOUNDARY ELEMENTS

For the calculation of the workspace the data of both manipulators and the load/cross-beam are required. In the workspace calculation the POI is determined first. In the next step the points are calculated for M1 and M2. At the end, the points of both manipulators are combined and hence, the points of the workspace boundary are given.

A. Parameters

The parameters enclose the manipulator data and the load/cross-beam. First, the manipulator data contain the maximum outreach depending on the load, the minimum outreach, the safety circle and the distance between both manipulators. The second parameter block load/cross-beam contains the length, center of gravity or the elective point on the load/cross-beam and the angle of the load/cross-beam.

B. Points of Manipulator M1

The calculation of the points of M1 occurs as depicted on the program flowchart in Fig. 6. In the first step the parameters from the file which are needed in the following steps are loaded. Then the points $P1$ to $P8$ are calculated, see Fig. 3. The branching out $\varphi < \alpha$ checks the angle α. The variable φ describes the angle of the load/cross-beam and α the angle of the tangent to the safety circle with radius r_{1s}, which goes through the center of manipulator M2. The

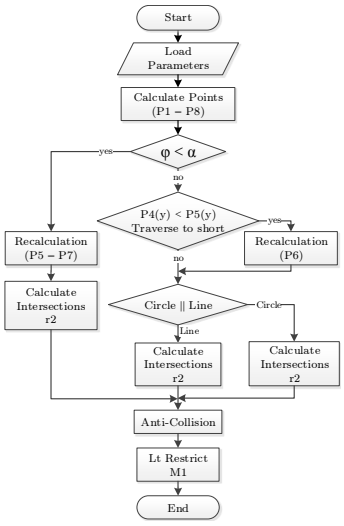


Fig. 6. Points of manipulator M1.

described variables are evident in Fig. 7. The angle can be calculated with the following equation:

$$\alpha = \arcsin\left(\frac{r_{1s}}{d}\right). \tag{1}$$

If the angle $\varphi < \alpha$, then the points $P5$ to $P7$ must be

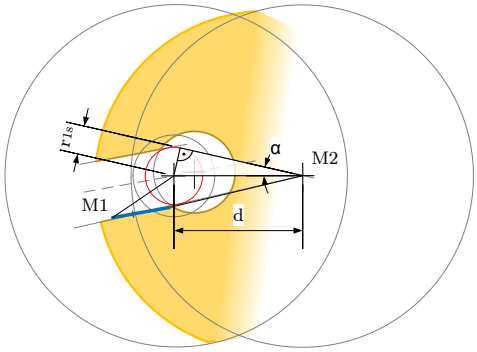


Fig. 7. Angle α.

calculated again, see Fig. 3. If the angle of the load/cross-beam $\varphi > \alpha$, then it must be verified if the load/cross-beam is not too short. This occurs if the y- value from $P5$ is smaller than $P4$; therefore $P6$ must be calculated again. With the next branching it is checked whether the maximum outreach of M2 intersects the safety circle r_{1s} or the line between $P4$ and $P5$. This takes into account the displacement of the circle center m_1 (Fig. 4). In Fig. 8 the required parameters are shown. If the length x_C is longer than the distance d between both manipulators, an intersection with the connecting line is possible. The length x_k is calculated with the following equation:

$$x_k = \frac{x_s}{\cos(\varphi)}. \tag{2}$$

In the boxes ”Calculate Intersections r2” the intersections with the maximum outreach of M2 are calculated. It is checked with which line element or circle segments the

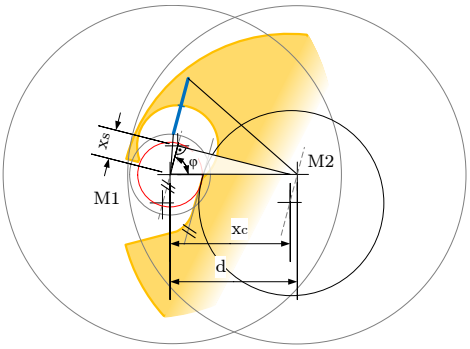


Fig. 8. Difference line - circle.

maximum outreach of M2 intersects, see Fig. 5. If all points are calculated, the workspace boundary must be checked for possible collision areas. After the operation ”Collision” the constraints must be checked. Every point of the workspace boundary is checked by M1 under the consideration of the load/cross-beam on their attachment point. If the attachment point of M1 is beyond the maximum outreach of M1, the point is calculated once more.

C. Points of Manipulator M2

The calculation of the points for M2 occurs after the same principle and procedure as with M1. The points are calculated by the input dimensions of M2. $P1$ and $P19$ are similar in their behavior, but they are in general different in the input dimensions, for instance, the outreach or the manipulator position. Conceptual the points of M1 can be turned $180°$ about a pivot pin to receive the workspace boundary of M2. This is only valid if both manipulators are precisely the same.

D. Points of the Workspace Boundary

The outcome of the calculation are the points for the workspace boundary. These points are for a load/cross-beam angle from $0°$ to $120°$. To receive the workspace boundary for negative load/cross-beam angles, the points must be reflected about the y-axis.

E. Workspace Separation

If the movement with the load/cross-beam through both manipulators is not possible anymore, it comes to a workspace separation. The additional points $P15$ and $P16$

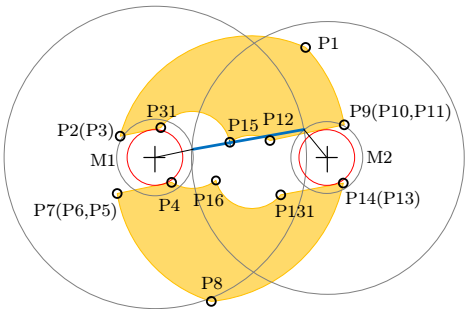
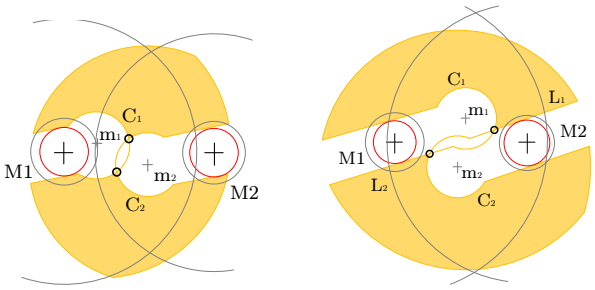


Fig. 9. Workspace separations.

which are needed for the workspace separation are illustrated in Fig. 9. To recognize a workspace separation, intersections need to be determined and checked. The intersections are shown in Fig. 10. If the circles C_1, C_2 in Fig. 10(a) or the



(a) Circles. (b) Lines and circles.

Fig. 10. Intersection of the workspace separation.

lines L_1, L_2 with the circles C_1, C_2 intersect, see Fig. 10(b), then a workspace separation is given. If the center of a circle $m_1(x) > m_2(x)$, then the circles change their sides. The marked points in Fig. 11 must be calculated.

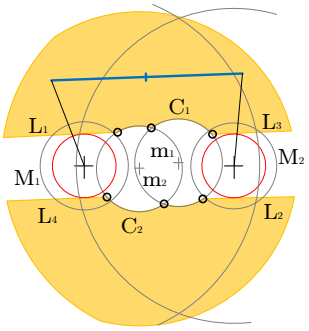


Fig. 11. Change of sides.

F. Collision

The physical boundaries of a manipulator do not allow moving with a boom over or under the other one. Due to this known collision areas, this must also be considered with the workspace boundary and finally has to be excluded. The collision area of M1 is shown in Fig. 12 with the points of the workspace boundary. To define the collision area, the

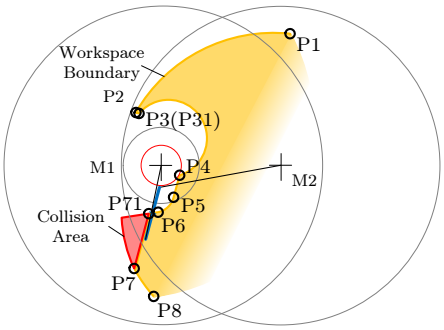


Fig. 12. Collision area.

point $P71$ must be introduced, in addition. The point $P7$ is checked for his angle in dependence of the base coordinate system. According to the size of the load/cross-beam, the angle φ must be distinguished:

$$\varphi_{Tr} = \begin{cases} \varphi & : 0 \leqslant \varphi \leqslant \frac{\pi}{4} \\ \varphi - \frac{\pi}{4} & : \varphi > \frac{\pi}{4}. \end{cases} \tag{3}$$

The angle φ_{P7} of the point $P7$ is calculated with the following equation:

$$\varphi_{P7} = \arctan\left(\frac{P7(y)}{P7(x)}\right). \tag{4}$$

To check whether a collision is possible, both angles are compared. If $\varphi_{Tr} > \varphi_{P7}$, then it comes to a collision. The point $P71$ must be calculated and $P7$ is shifted (Fig. 12).

IV. MERGING OF THE BOUNDERY ELEMENTS

After calculation of all points for M1 and M2, they will be merged as shown in Fig. 13 and finally the topology of the workspace is checked and plotted via a 3D plot.

A. Merging the Points

In the first step the parameters from the file are loaded. They contain information for merging the single points. If a restriction of M1 and/or M2 occurs, the points are calculated again. A restriction is not given if the points are congruent. For M1 these are the points $P1, P21$ and for M2 the points $P1, P141$ and $P8$. In the next box the points are checked for

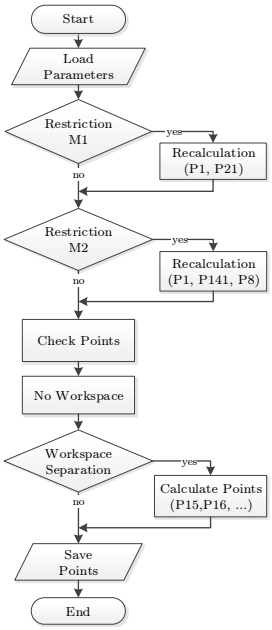


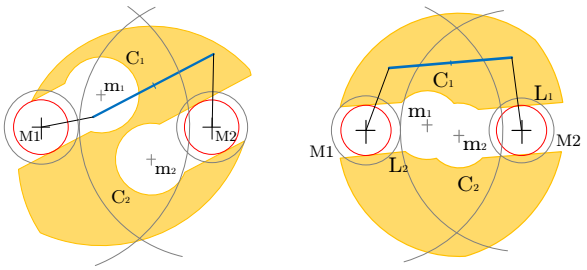
Fig. 13. Merging points.

correctness and are changed if they are not correct. The major task of the control lies at very high load/cross-beam angles. If no workspace exists, the points are put on the coordinates $(0, 0)$. This concerns all points of the workspace boundary. The branching "Workspace Separation" checks whether it comes to a possible workspace separation. If the movement

with the load/cross-beam through both manipulators is not possible anymore, it comes to a workspace separation. New points ($P15, P16$) are defined and existing points are recalculated.

B. Connected / Disconnected Workspace

Depending on the angle and length of the load/cross-beam, the workspace can have different topologies, either one closed/connected or two separated regions, see Fig. 14. The



(a) Connected. (b) Disconnected.

Fig. 14. Workspace.

critical points, where the topology changes can be computed via:

$$C_1: \quad (x - x_s\cos(\varphi))^2 + (y - x_s\sin(\varphi))^2 = r_{1m}^2 \tag{5}$$

$$C_2: \quad (x - (L_t - x_s)\cos(\varphi))^2 + (y - (L_t - x_s)\sin(\varphi))^2 = r_{2m}^2 \tag{6}$$

$$\arg\min_{\varphi}(\{(x,y) : C_1(x,y) = C_2(x,y)\} = \emptyset) \tag{7}$$

where x, y are the coordinates of the circle centers. The formula is valid for angles from $\varphi = 0°$ to $120°$. To find the solution for $\varphi = 0°$ to $-120°$, take the mirror image about the y-axis from before.

C. 3D-PLOT

In addition to the 2-dimensional plot of the workspace a third dimension is added, the load/cross-beam angle φ. The

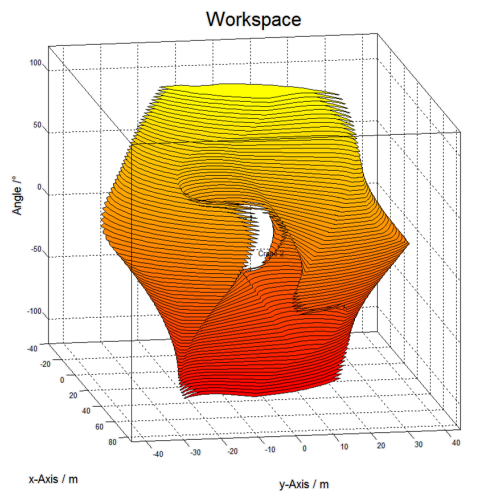


Fig. 15. 3D-Plot.

angle φ is varied within the range of $\pm 120°$. The plot holds for the following set:

$$\varphi = \{k \cdot 3 | k \in \mathbb{Z} \wedge -40 \leq k \leq 40\}. \tag{8}$$

The hole represents the disconnected workspace. If there is no hole, then it is always possible to move through both manipulators for any angle within $\pm 120°$. Another three-

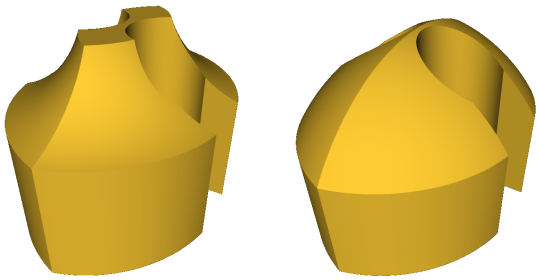


Fig. 16. Three-dimensional LML and workspace.

dimensional plot is possible for a fixed angle φ for the LML (load moment limitation) and the workspace of the POI including the possible maximal height see Fig. 16.

V. MAXIMUM CROSS-BEAM LENGTH

The maximum cross-beam length (e.g. length between hooks) describes the largest distance where a motion through both large scale manipulators is possible. There are two different positions for the load possible, which are the two critical positions, see Fig. 17. Only with the shorter length it is possible to move from one side to the other. The longer length can move up to a certain degree between the manipulators, however not further, since the other shorter one is the limiting factor during a motion through both manipulators. The maximum length can be computed via:

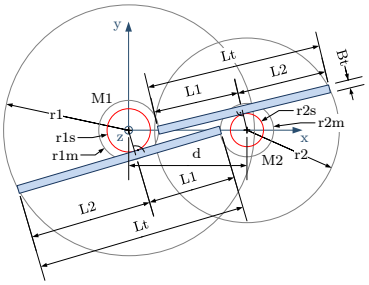


Fig. 17. Maximum cross-beam length.

$$L_1 = \sqrt{(d - r_{jm})^2 - \left(r_{ks} + \frac{B_t}{2}\right)^2} \tag{9}$$

$$L_2 = \sqrt{r_k^2 - \left(r_{ks} + \frac{B_t}{2}\right)^2}, \tag{10}$$

and finally:

$$L_{max} = \min_{j,k}(L_1 + L_2), \tag{11}$$

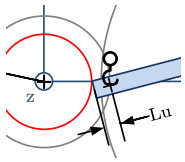


Fig. 18. Side extension of the load/cross-beam.

with $j \in \{1,2\}$, $k \in \{1,2\}$, $j \neq k$. In the case that the load extends (e.g. the hook anchor points) by a distance L_u, see Fig. 18, then the maximum side extension is calculated via:

$$L_{1min} = \min_{j,k}(L1) \tag{12}$$

and finally as:

$$L_u = L_{1min} - \sqrt{(d - r_{js})^2 - (r_{ks} + B_t)^2}, \tag{13}$$

with $j \in \{1,2\}$, $k \in \{1,2\}$, $j \neq k$. If $L_u < 0$, then the length L_u needs to be subtracted from the maximum length L_{max}.

VI. APPENDIX

In the appendix same mathematical methods are described, which are needed for the computation of the workspace.

A. Chordal Lines

The intersection of two circles can be computed with the help of chordal lines (dt.: Potenzgerade, Potenzachse) [12]. The chordal line is the set of all points where the exponent of both circles is equal. It is orthogonal to the line connecting the origins of the circle. For equal radii it is equal to the bisecting line, see Fig. 19. The chordal line is not defined for concentric circles. The general form of the circle equations

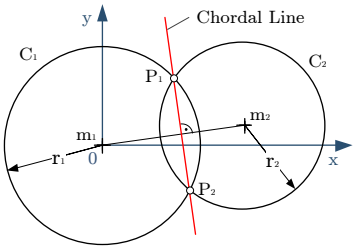


Fig. 19. Chordal line.

for the circles C_1 and C_2 are (14) and (15) as follows:

$$\text{C1:} \quad (x - x_{m1})^2 + (y - y_{m1})^2 = r_1^2 \tag{14}$$

$$\text{C2:} \quad (x - x_{m2})^2 + (y - y_{m2})^2 = r_2^2 \tag{15}$$

Subtracting both equations gives the equation of the chordal line (16):

$$y = \underbrace{\frac{x_{m1} + x_{m2}}{y_{m1} - y_{m2}}}_{k} x + \underbrace{\frac{x_{m2}^2 + y_{m2}^2 - r_2^2 - x_{m1}^2 - y_{m1}^2 + r_1^2}{2\,(y_{m1} - y_{m2})}}_{d}. \tag{16}$$

Substitution of the chordal line (16) into a circle equation (14) solve for x and then plug into the chordal equation(16) to achive the intersection points of the circles. Substitution into the circle equation would give two solutions, where only one is valid.

B. Tangent Line at a Circle

To compute the touching point point of a tangend line to a circle (Fig. 20) intersect a circle C and a line L. Given is the circle equation and the slope of the line. Substitute the

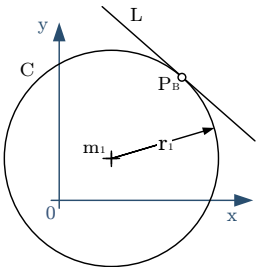


Fig. 20. Tangent line.

circle equation (17) into the line equation (18) and solve for y to get the intersection point (19).

$$\text{C:} \quad (x - x_{m1})^2 + (y - y_{m1})^2 = r_1^2 \tag{17}$$

$$\text{L:} \quad y = k\,x + d \tag{18}$$

$$y = \frac{-k\,d + x_1 + k\,y_1}{1 + k^2} \pm \frac{\sqrt{-2kdx_1 + 2x_1ky_1 - y_1^2 + r_1^2 + 2dy_1 - d^2 + k^2r_1^2 - k^2x_1^2}}{1 + k^2} \tag{19}$$

The tangent line has only one point in common, a so called double point with the circle. Therefore the expression within the square root needs to be equal to zero. Hence, the following expression can be found for d.

$$d = -x_1\,k + y_1 \pm \sqrt{r_1^2 + k^2\,r_1^2} \tag{20}$$

Plugging in the solution into Eq. 20 into the line equation (18) and then into the circle equation (17) gives the intersection point P_B.

REFERENCES

[1] International Organization for Standardization, ISO 12480-1: Cranes Safe Use, Part 1: General, 1997.
[2] Eberharter, J. K. and Schneider, K., Tandem Control, Yokohama, 2009.
[3] Eberharter, J. K., Rajek, M. and Schneider, K., Synchronisierte Mehrkranhübe, Internationales Forum Mechatronik, 2009.
[4] Vaughan, J., Yoo , J., Knight, N. and Singhose, W., Multi-Input Shaping Control for Multi-Hoist Cranes, Amer. Control Conf., 2013.
[5] Hamilton, M.R., Preplanning Analysis and Computer Animation of Dual Crane Lifts in Highway Construction, thesis, The University of Texas at Austin, 1992.
[6] Haas, C. and O'Conner, J., Computer aided critical construction operations, 8, 1999.
[7] Varghese, K., Dharwadkar, P., Wolfhope, J. and O'Conner, J.T., A Heavy Lift Planning System for Crane Lifts, Microcomputers in Civil Engineering, **12** p. 31-42, 1997.
[8] KranXpert, http://www.kranXpert.com.
[9] Liftplaner software, http://www.liftplanner.net.
[10] Liebherr Crane-Planner, http://www.crane-planner.com.
[11] Kaufmann, G., Analyse des Arbeitsraumes für den Tandemhub, Thesis, University of Applied Sciences, Dornbirn, Austria, 2012.
[12] Coxeter, H.S.M. and Greitzer, S.L., Klett-Studienbücher, Klett, ISBN=9783129833902, 1983.

Innovative concepts in educational robotics: Robotics projects for kindergartens in Austria

Johann Eck[1], Sabine Hirschmugl-Gaisch[2], Alexander Hofmann[4], Martin Kandlhofer[3], Sabrina Rubenzer[4] and Gerald Steinbauer[3]

***Abstract*— Using robotics platforms for kindergarten children to interest them for computer science is a rather new idea in educational robotics. In this paper we present two different robotics projects in kindergartens. The Department of Computer Science at the University of Applied Sciences Technikum Wien established a course setting for children aged four to six years. Within the project they compared the performance of those kindergarten kids who received the robotics workshop with those who did not attend the training in order to analyze the effects of the course on the children's cognitive processes. The second project, initialized by the Graz University of Technology and the University of Teacher Education Styria, focuses the cross-generational aspect involving kindergarten kids, pupils as well as senior citizens. Within the project a robotics day in a kindergarten offering eleven different hands-on experiments for children was organized and a first qualitative feedback was obtained.**

I. Introduction

Educational robotics has gained increased attention in the last decades. Several conferences and workshops deal with the use of robotics in education [21]. Initiatives like *RoboCupJunior (RCJ)* aim to interest young children and pupils up to the age of nineteen in science and technology [25]. On the contrary educational robotics with special focus on children aged between three and six years is less widespread. Science and technology are changing rapidly and young children have to be prepared for this development. The idea behind the concept of educational robotics in kindergarten is to use the robot as pedagogical tool to familiarize children in pre-school age with science and technology in a playful way. By presenting two innovative projects for kindergartens in Austria this paper discusses how different robotics platforms could be integrated in the education of children between three and six years of age. Furthermore, it presents first results of an empirical study evaluating the effects of using robotics in kindergarten.

The remainder of the paper is structured as follows: Chapter II deals with related research whereas chapter III gives a brief overview of the current situation of educational robotics in kindergartens in Austria. Chapter IV provides a detailed description of the kindergarten projects followed by the presentation of preliminary results aiming at this goal in chapter V. Chapter VI discusses conclusions and future work.

*Authors listed in alphabetical order.

[1]J. Eck is with the University of Teacher Education Styria, Austria hans.eck at ainet.at

[2]S. Hirschmugl-Gaisch is with the University of Teacher Education Styria and the Kindergarten Rosental a.d. Kainach, Styria, Austria hirschmugl.gaisch at aon.at

[3]M. Kandlhofer and G. Steinbauer are with the Institute for Software Technology at Graz University of Technology, Austria [mkandlho, steinbauer] at ist.tugraz.at

[3]A. Hofmann and S. Rubenzer are with the University of Applied Sciences Technikum Wien, Austria [alexander.hofmann, sabrina.rubenzer] at technikum-wien.at

II. Related research

As the level of awareness and importance of educational robotics rose over the last decades a great number of conferences, workshops, papers and books address this topic [20], [3], [21]. Alimisis and colleagues [1] for instance provide in their book an extensive overview of the theoretical background as well as practical aspects of robotics in education.

The paper in [23] describes how robotics can act as a tool to teach pupils the basics of engineering and programming. In addition they conducted empirical studies in order to investigate why robots seem to motivate children, even if they were not technically interested beforehand.

Whereas the use of robotics in pre-school education is not as wide-spread as in primary and secondary school various papers and articles exist which describe robotics platforms and projects for young children. For example the authors of [6] present the experiences made introducing robotics in a kindergarten using *Lego WeDo*. Children had to build a small robot step by step. Afterwards they interacted with the robot, which was actually programmed by a teacher.

The article in [2] describes the integration of robotics in early childhood education following a constructionist strategy (learning by designing, using concrete objects to explore, identification of powerful ideas, self-reflection [2]).

Janka [15] presents the use of the programmable robot-toy *Bee-Bot* in pre-school education. Different activities and games for kindergarten children and teachers were designed and qualitatively evaluated. The focus of this research was based on robot programming instead of construction and design. It turned out that although all children involved in the study basically enjoyed playing with the Bee-Bot and were not afraid of using this new technology the robot itself was not interesting to them for a longer period of time. The author also states that some of the children showed a basic understanding of the robot's control principles whereas others seemed to be too cautious to increase their self confidence during the work with the Bee-Bot [15].

III. Current status

Educational robotics for primary and secondary schools is well established in Austria. Among other initiatives a nationwide network of RoboCupJunior regional centers provides

support for schools, teachers and pupils [13]. On the contrary only a few initiatives and projects can be found which use robotics in kindergarten and pre-school education.

One example would be the robotics course "Robots for Kids". In 2010 the Department of Computer Science at the University of Applied Sciences Technikum Wien set up this course in cooperation with "Kinderfreunde Wien". The target group for this course are kindergarten children at the age of four to six years.

As another example the project "Technical and natural science in playschool" of Vienna University of Technology could be mentioned. Children aged between four and six have the opportunity to visit different departments of the university and participate in experiments. Within this project one of the main topics is robotics.

Additionally, different scientific institutions and universities offer training courses and workshops for educators and children. For instance the Austrian Computer Society offers robotic workshops in order to teach kindergarten pedagogues how to integrate robotics into teaching. The "Technisches Museum Wien" organizes workshops for children between the age of four and seven to teach them the basics of programming and robotics.

Fig. 1. Two children working with the *Bee-Bot*

The initiative "Children visit Science" is an innovative approach within the context of kindergarten pedagogy in Austria. The intergenerational, cross-organizational project was originally initiated in 2010. The basic aim of this initiative is to provide pre-school children and pupils with access to different scientific fields and furthermore to give an insight into the research sector at different scientific institutions [14], [12].

In the first year the initiative comprised five educational modules, focusing on different topics (bioscience, experimental physics, criminalistics, chemistry, paper manufacturing). In spring 2012 a scientific project day on the subject of electrostatics and electricity was organized. Secondary school students in cooperation with their teachers prepared different hands-on experiments. Pupils acted as guides explaining the experiments to kindergarten children. This concept formed the basis of the scientific robotics day described in section IV-B [5], [11], [14], [12].

Almost all above mentioned robotics projects and workshops use the *Bee-Bot*, manufactured by the British company *PrimaryICT*, as a learning tool (see Figure 1). The small programmable wheeled robot, designed for pre-school and lower primary school children, is a widely adopted tool within the context of educational robotics in kindergarten. It can be controlled according to the principles of the *Logo* programming language [22]. Using the buttons on the back of the robot (forward, backward, rotate left, rotate right) children can enter a sequence of commands. Each forward/backward instruction moves the robot 15cm in the corresponding direction whereas each rotation instruction turns the robot by 90 degrees without changing its current position [15].

IV. Project Description

The two robotics projects for kindergartens presented in this paper were carried out in Vienna and Graz. The following subsections provide a detailed description of each project.

A. Evaluation of the robotics course "Robots for Kids" for kindergarten children

The idea of evaluating this course arose while questioning whether the settings and the used contents are reasonable and helpful to teach first principles in programming using robots as a learning environment.

As a consequence it was decided to redesign the course in order to accomplish a setting, which can be evaluated by students of the Faculty of Psychology at the University of Vienna during the period from March to June 2012.

Therefore, the goal was to analyze the effects of the course on children's cognitive processes in the context of executive functions. These are important parts when coping with the everyday school routine and are utilized for planning and initiating actions. The main issue was to evaluate whether the training with the robots has an influence on the performance of the executive functions [28].

Settings and content: There have been four kindergartens (private as well as public) involved in the project. Two of them with a group size of 12 to 16 children participated in the course, whereas the other two kindergartens with a group size of 27 children in total have not attended the course and have been used as a control group in order to compare the results. As a condition to participate in the project, the minimum age of the children was defined with 54 months (four-and-a-half-years). An absence of more than two lessons led to an exclusion of the target group.

The training was divided into six units, which were held in the morning at weekly intervals and lasted one hour. Four students of the University of Applied Sciences Technikum Wien executed the redesigned and standardized training schedule. Because of their simple user interfaces and the possibility to utilize them in team constellations, Bee-Bot robots have been chosen as training objects. The tasks which should be fulfilled each training have been defined

in advance in order to guarantee that the two kindergartens have been subjected to the same terms. Additionally, the students recorded on a prepared sheet how many children were able to solve the specific tasks to get an overview of the performance of the kids.

Aims and hypotheses: The aim was to compare the performances of the pre-school children before and after the training. Therefore, four hypotheses were formulated [28]:

Hypothesis 1. The performances of the children who attended the robot course have improved in the post-test compared to those of the pre-test. If the result is significant it has to be worked out whether the increase can be explained by the robotics training. Therefore, the results have to be compared with those of the control group (children who were tested without having attended the training).

Hypothesis 2. After attending the training the performance of the executive functions is significantly better in the test group than in the control group.

Hypothesis 3. The statistical connection between the variables inhibition, shifting and planning is significant.

Hypothesis 4. Demographic factors have an influence on the performance at the pre-test.

B. A cross-generational robotics project day in kindergarten

In November 2012 a scientific kindergarten experiment day with special focus on robotics was organized as a joint project between the New Secondary School Voitsberg, the Kindergarten Rosental a.d.Kainach (both in Styria), the University of Teacher Education Styria and Graz University of Technology (TUG). The structure of the robotics day was based on the concept "Children visit Science" and the scientific project day on electrostatics and electricity described in section III.

One main objective of the robotics project day was to prepare contents of the area of robotics respecting pedagogical and didactic aspects as well as principles of educational robotics ([7], [27], [24], [1]). Therefore, members of the robotic lab at TUG together with kindergarten pedagogues and teachers developed eleven different hands-on experiments and educational games applying methods of research-based learning ([19]) and the technique of story-telling ([14], [17]). Respecting fundamental principles of educational robotics as stated by Frangou and colleagues in [7] children could actively participate, explore, test and interact with the robots.

During the project day at the kindergarten each experiment was carried out at a separate hands-on area, also referred to as 'experiment station'. According to the concept of an education partnership ([26]), secondary school students carried out and explained the experiments to kindergarten children and their grandparents. Pupils slip into the part of a teacher, accompanying the kindergarten children through their way of discovering and experiencing. In preparation for their tasks pupils attended a half-day robotics workshop. In this workshop the young students were first introduced to the basic concepts of robotics and the scientific background of each robotics experiment (e.g. explanation of sensor, motors, robot programming, and so forth). Afterwards they got detailed instructions on how to carry out and guide the different experiments. In parallel kindergarten pedagogues plan and carry out extensional framework-projects in order to prepare the pre-school children.

To give the different age groups participating (pre-school children, pupils, senior citizens) a basic understanding of robotics and artificial intelligence the experiment stations were structured around following major items using different robotics platforms: The programmable wheeled robot *Bee-Bot* [15], functionality of sensors using the *LEGO Mindstorms NXT 2.0* robotic kit [16], the humanoid robot on the example of the *Hitec RoboNova* [10] and finally mapping and object tracking using the *Pioneer 3 DX* robot [9]. Figure 2 shows the excitement of children and pupils at the different stations. Following a short description of each topic.

1) Telling a story using the Bee-Bot: Based on the functionality of the Bee-Bot described in chapter III two educational games were developed. In the first game children had to program the robot to follow a certain path on a special square grid mat. The path represented the different production stages in a glass factory (also see Figure 1). The research question to the children was: "Can you teach the Bee-Bot how to make glass?". The task of the second game was to program the robot so that it moves from a starting point to an endpoint, stopping at certain intermediate positions on a square grid mat with fairy-tale motifs imprinted. The research question for this task was: "Can you tell the story of the *bad wolf and the three little piglets* whereby the Bee-Bot is acting the wolf?"

2) Functionality of sensors: Seven hands-on experiments demonstrated the use and the functionality of the ultrasonic-, the light-, the sound- and the color-sensor. Children could interact with the different robots which were build using Lego Mindstorms. Research topics included: "Follow the light", "Don't drop from the table", "Avoid collisions", "Sweet-serving service robot" (Figure 2c), "Find and grab the can", "Sort the color bricks" (Figure 2a) and "Follow the noise".

3) Humanoid robots: Using the example of the RoboNova the basics of humanoid robots were demonstrated. Pupils could control the robot by sending commands via the infrared remote controller. Children had to watch the robot carefully and afterwards imitate its movements (Figure 2b). The research question was: "Is a robot better at dancing than me?"

4) Mapping and object tracking: This experiment station dealt with the topics of mapping and object detection using the Pioneer 3 robot with a SICK laser scanner and a Microsoft Kinect camera (Figure 2d). The tasks for the children were formulated as follows: "Supporting the rescue robot" and "Playing football with a real robot"

(a) Brick sorter (b) Humanoid dancing robot (c) Service robot (d) Rescue robot

Fig. 2. Kindergarten children and pupils together carrying out hands-on robotics experiments

V. Preliminary evaluation and results

This section describes the methodology used and results of the evaluation of the robotics course "Robots for Kids". Subsequently the outcome and preliminary qualitative evaluation results of the cross-generational kindergarten robotics project day will be presented.

A. "Robots for Kids": Evaluation methodology and results

Both the test group as well as the control group have been pre- and post-tested. The pre-tests took place in March and April 2012 with a time gap of one week minimum before the training started and the duration was announced with about 50 minutes (with a break after the first 20 minutes) for each participant.

To operationalize the second hypothesis a mixed design analysis with repeated measurements was chosen. Independent variables are the variables of the test- and control group as well as the two time measurements of the pre- and post-test. The dependent variables have been constituted by the performances of the children at the tests to determine the executive functions.

For the pre-test and post-test, following psychological instruments have been chosen:

- Kaufmann Assessment battery for Child [18]: displays the intelligence level and the language skills of the children. This instrument was just used for the pre-test.
- Dimension change card sorting test [29]: determines the shifting (cognitive flexibility), which means to display if the children are able to apply to newly learned rules.
- Day-Night-Stroop [8]: measures the inhibition respectively the endurance and the ability to concentrate over a certain period of time.
- Truck Load [4]: used to work out if the participants are able to plan their next steps (pushing the right button of the Bee-Bot).

After the six-week training the post-tests have been executed from May to June 2012. These have been conducted by other students of the Faculty of Psychology to avoid an influenced testing effect and the duration was 20 minutes.

The outcome of the evaluation of the robotics workshop for kindergarten children can be summarized as followed.

Regarding the performance of the test-group it could be figured out that there have been improvements in the area of planning and cognitive flexibility, but these have not been significant. However, the efforts concerning inhibition were significant. As a result it can be stated that the performance of the children who attended the robot course has been improved in the field of endurance and the ability to concentrate over a certain period of time.

On the contrary it could not be proved that performance after the training in the test group is significantly better than in the control group. The improvements have been nearly equally. Nevertheless this could be affiliated by a learning effect caused by the use of the same testing instruments at the pre- and post-test.

The third hypothesis can be proved right; there is a statistical connection between the three variables. This implicates that they can be matched with the construct of executive functions. An improvement in one of the areas (inhibition, planning, or cognitive flexibility) could therefore lead to an improvement in one of the others as well.

Concerning the demographic factors, simply the level of education of the participant's parents had a significant effect on the different performances (of the test group in comparison to the control group) at the pre-test in the field of planning.

Summing up, the results do not show statistically significant improvements (despite the field of inhibition). Reasons for that could be affiliated by the used tests. It is possible that the tests have not been sensible enough to show the changes of the executive functions caused by the robotic training [28].

B. Kindergarten robotics day: Outcome and preliminary results

Respecting pedagogical and didactic aspects the first cross-generational robotics day was conducted. In sum twenty-five kindergarten children, divided into groups of three, and ten pupils participated. Each group of children was accompanied by at least one grandparent. The described approach combined two major benefits: On the one hand pupils learned about scientific topics not only during the preparation process but also afterwards by guiding and explaining the experiments to kindergarten children. On the other hand kindergarten children had the opportunity to

learn and gather practical experiences together with pupils and senior citizens. In this context one important aspect was that pre-school children could actively participate in the experiments. Furthermore the integration of different age groups and different educational institutions fostered a vital social process between kindergarten children, young students, senior citizens as well as mentors, teachers and staff members of participating institutions. In general the concept of discovering and experimenting represents a valuable pedagogical approach within the area of pre-school education, fostering the learning process of children in a holistic way. In addition the robotics day formed the basis for a follow-up project at the kindergarten in order to deepen what children have seen and experienced [14], [12].

During the robotics day pictures were taken and experiments were videotaped to gather qualitative data. Considering ethical and legal aspects all collected data was treated confidentially. Beforehand parents were informed and asked for their permission to take pictures and to videotape experiments. Gathered data is still being analyzed, findings will be published and discussed at a later date.

Right after the robotics day qualitative feedback from kindergarten pedagogues, grandparents, parents and children was obtained. This feedback was mainly positive. For instance some parents reported that both children and their grandparents were motivated to build robots on their own after participating in the robotics day (i.e. using Lego Mindstorms). One teacher told about a child with special needs which also participated in the robotics day. The day after both the child's occupational therapist and psychologist noticed a significant improvement of it's behaviour. Kindergarten pedagogues reported that children were very enthusiastic about their first robotics-experience and still, almost half a year later, asking when the robots will return.

Science and technology develop rapidly. In order to prepare children it is important to familiarize them already in kindergarten with science and technology in a playful way. As first comments and qualitative feedback from the robotics day indicate, using robots as pedagogical tools could be one way to achieve this goal.

VI. Conclusions and Future work

In this paper two concepts of integrating robotics in kindergartens in Austria have been presented. Furthermore, quantitative results of an evaluation of a robotics workshop for kindergarten children as well as preliminary qualitative evaluation results of a cross-generational kindergarten robotics day have been discussed.

A future evaluation project will use different psychological instruments for pre- and post-testing while evaluating a robotics course in order to avoid learning effects. Moreover, other tests, which measure the competencies used in robotics training in a more detailed way, could be applied [28]. In addition effects of robotics courses on emotional understandings can possibly be displayed. This is currently being analyzed.

In order to refine and improve the contents of the kindergarten robotics day presented in this paper qualitative interviews with participating children, pupils and teachers will be conducted. Based on the findings of those interviews and the lessons learned from the first robotics day further project days in different kindergartens in Styria will be organized. In addition a more detailed quantitative and qualitative evaluation on the impact of such robotics days in kindergartens is planned.

Acknowledgements

The work has been partly funded by the European Fund for Regional Development (EFRE), the federal government of Slovenia and the Land Steiermark under the Tedusar grant. The robotic day at the Kindergarten Rosental was a module from the series "Children visit Science" in cooperation with the Interdisciplinary Center for Teaching Methodology at the University of Teacher Education Styria

References

[1] D. Alimisis and C. Kynigos. Constructionism and robotics in education. In D. Alimisis, editor, *Teacher Education on Robotics-Enhanced Constructivist Pedagogical Methods*. School of Pedagogical and Technological Education (ASPETE), 2009. ISBN 978-960-6749-49-0.

[2] M. U. Bers, I. Ponte, C. Juelich, A. Viera, and J. Schenker. Teachers as designers: Integrating robotics in early childhood education. *Information Technology in Childhood Education*, 2002(1):123–145, 2002.

[3] A. Bredenfeld, A. Hofmann, and G. Steinbauer. Robotics in education initiatives in europe - status, shortcomings and open questions. In *International Workshop 'Teaching robotics, teaching with robotics', SIMPAR 2010*, Darmstadt, Germany, November 2010.

[4] S. Carlson, L. Moses, and L. Claxton. Individual differences in executive functioning and theory of mind: An investigation of inhibitory control and planning ability. *Experimental Child Psychology*, 87:299–319, 2004.

[5] H. Eck and W. Gaggl. IMST Regionales Netzwerk Steiermark, Bericht 2011/2012. Final annual report, 2012.

[6] F. Ferreira, A. Dominguez, and E. Micheli. Twitter, robotics and kindergarten. In M. Moro and D. Alimisisi, editors, *Proceedings of 3rd International Workshop 'Teaching robotics, teaching with robotics. Integrating Robotics in School Curriculum*, Riva del Garda, Italy, April 20 2012.

[7] S. Frangou, K. Papanikolaou, L. Aravecchia, L. Montel, S. Ionita, J. Arlegui, A. Pina, E. Menegatti, M. Moro, N. Fava, S. Monfalcon, and I. Pagello. Representative examples of implementing educationa robotics in school based on the constructivist approach. In *Workshop Proceedings of SIMPAR 2008; Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS*, pages 54–65, Venice, Italy, November 2008.

[8] C. Gerstad, Y. Hong, and A. Diamond. The relationship between cognition and action: performance of children 3 1/2-7 years old on a stroop-like day-night test. *Cognittion*, 53:129–153, 1994.

[9] G. Grisettiy, C. Stachniss, and W. Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005.

[10] D. Grunberg, R. Ellenberg, Y. E. Kim, and P. Y. Oh. From robonova to hubo: Platforms for robot dance. *Progress in Robotics; Communications in Computer and Information Science*, 44:19–24, 2009.

[11] S. Hirschmugl-Gaisch and H. Eck. *Von magischen Koepfen und leuchtenden Wanzen*. 2012.

[12] S. Hirschmugl-Gaisch, H. Eck, and H. Jungwirth. Kinder reisen durch die wissenschaft. In *Fachtagung fuer elementare Bildung*, Graz, Austria, September 2011.

[13] A. Hofmann and G. Steinbauer. The regional center concept for robocupjunior in austria. In *First International Conference on Robotics in Education*, Bratislava, Slovakia, 2010.

[14] IMST. Kinder reisen durch die Wissenschaft. Report, 2011. Report by the IMST initiative (Innovations make schools top).

[15] P. Janka. Using a programmable toy at preschool age: Why and how? In *Teaching with robotics: didactic approaches and experiences. Workshop of International Conference on Simulation, Modeling and Programming Autonomous Robots (SIMPAR 2008)*, 2008.

[16] S. H. Kim and J. W. Jeon. Programming lego mindstorms nxt with visual programming. In *International Conference on Control, Automation and Systems*, Seoul, Korea, October 2007.

[17] S. Masemann and B. Messer. *Improvisation und Storytelling in Training und Unterricht*. Beltz, 2009.

[18] P. Melchers and U. Preuss. *Kaufmann Assessment Battery for Children*. Frankfurt/Main: Pearson Assessment, 2009.

[19] R. Messner. *Schule forscht: Ansaetze und Methoden zum forschenden Lernen*. edition Koerber-Stiftung, 2009.

[20] M. Moro and D. Alimisisi, editors. *Proceedings 3rd International Workshop 'Teaching robotics, teaching with robotics'*, Riva del Garda, Italy, April 20 2012.

[21] D. Obdrzalek, editor. *Proceedings of the 3rd International Conference on Robotics in Education (RiE 2012)*, Prague, Czech Republic, September 2012.

[22] S. Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, 1993.

[23] M. Petre and B. Pricer. Using robotics to motivate back door learning. *Education and Information Technologies*, 9(2):147–158, 2004.

[24] E. Romero, A. Lopez, and O. Hernandez. A pilot study of robotics in elementary education. In *10th Latin American and Caribbean Conference for Engineering and Technology*, Panama City, Panama, July 2012.

[25] E. Sklar. A long-term approach to improving human-robot interaction: Robocupjunior rescue. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*, 2004.

[26] M. Textor. *Bildungs- und Erziehungspartnerschaft in Kindertageseinrichtungen*. Books on Demand, 2011.

[27] M. Virnes and E. Sutinen. Topobo in kindergarten: educational robotics promoting dedicated learning. In *Proceedings of the 17th International Conference on Computers in Education*, Hong Kong, November 2009.

[28] L. Wulf. Effects of Robotics. Master's thesis, University of Vienna, Austria, 2012.

[29] P. Zelazo, D. Frye, and T. Rapus. An age-related dissociation between knowing rules and using them. *Cognitive Development*, 1:37–63, 1996.

A ROS and Aria based framework for didactical analysis of behavioral control in mobile Robotics

Mario Grotschar Departments of Mechatronics
University of Applied Sciences Technikum Wien
Hoechstaedtplatz 5, 1200 Wien, Austria, Europe
mario.grotschar@technikum-wien.at

Clemens Doppler Departments of Mechatronics
University of Applied Sciences Technikum Wien
Hoechstaedtplatz 5, 1200 Wien, Austria, Europe
clemens.doppler@technikum-wien.at

***Abstract*— Due to the ever increasing complexity of software development in the field of robotics, so called "Robotic Development Environments (RDE)" constantly gain importance [8]. Falling in this category the "Robot Operating System (ROS)" from Willow Garage, Inc. [13] allows an abstract view of robotic problems. Above that ROS can be considered a "Meta Operating System" that provides tools and methods to simplify the creation of robotic applications.**

The primary goal of this work is to supply students with the benefits of this "Meta Operating System" together with the means of creating applications for autonomous robot behavior. Based upon the three-tiered architecture described by [5] and [6] a framework for the behavioral control layer is presented in this work. Because all other layers of a control architecture for more complex robots are building on this layer, it is crucial for students to get a deeper insight and understanding of such a system. The best way of achieving this, is the possibility to experiment freely on this topic. Therefor this work presents a full package for doing so. By overcoming the need for caring about robot program structure, odometry, sensor data acquisition, or motor control, students can address their whole attention to small code-blocks for developing new behaviors and parameterizing the framework. For testing and verification of the created behavioral control, the package of the framework also comprises a simulator and a software compatible mobile robot platform. Concluding, an exemplary application of the framework is given in this paper.

I. INTRODUCTION

One purpose for research in mobile robotics is the use of autonomous robots in the field of service applications. These service applications, like e.g. guiding people through buildings [1], or transport of items [2], [3], [4], mostly happen to take place in unstructured environments including dynamic changes and obstacles. Due to this fact, mobile robots have to be able to cope with all of these influencing factors in nearly real-time. Only if the robot control architecture allows the robot to react appropriate to every likely situation it is able to act really autonomous. According to [5], [6] one approach for a capable control system is the three-tiered architecture (3T). The schematic diagram in Figure 1 shows the basic idea behind the 3T model. There are three layers which have to manage different control tasks. Each of these tiers does only directly interact with neighboring tiers. The robot hardware consists of the body, actors and sensors. As [7] stated in his work, the differentiation between the tiers can be seen in a spatial scope as well as in a time related scope. This means the closer a layer is connected to the robot hardware, the closer the vicinitiy is, it's considering. Accordingly, the transient time for control tasks needs to be shorter for lower layers. This issue can also be seen as a matter of knowledge. The higher a tier is located in den 3T model, the more knowledge it needs to fulfill its task. E.g. for path planning there is a need for a global map and the robots latest pose whereas for collision detection current sensor information is sufficient.

The base tier, so called behavioral control, including the communication with actor motion control (PID), serves as a direct interface to the robot hardware. It controls the movement of joints and receives data from intrinsic and extrinsic sensors. Furthermore it contains a set of behavior functions that drive robot actions and shall lead to a more robust execution of tasks in ambiguous situations. E.g. if the robot has to move around in an unstructured environment it needs to have a collision avoidance behavior and a behavior that moves the robot in a general direction. Behaviors work similar to human reflexes. There is no time-consuming reasoning but an instantaneous reaction of the robot - like a blink of the human eye. Predefined rules or functions determine a direct actor control output in a short loop according to current sensor readings and high level routine data. If there are two or more behaviors working in parallel, there also has to be a merging function that fuses the control output of all active behaviors in a reasonable way. A different set of behaviors can be used for different situations.

The middle tier, called Executive, acts like a mediator between behavioral tier and planning tier. It receives global goals from the upper layer, reports the current status of the robot to the upper layer, and decomposes the global plan into sets of subtasks that can be managed by reactive behaviors. It controls which behaviors or which sequence of behaviors is needed to achieve the goal. In [5] and [7] this layer is also referred to as tactical intermediate planner. It may manage a temporary local map that comprises dynamic changes in the environment. The executive layer may also recognize failures in a current maneuver and handle a recovery action. Depending by the implementation of the architecture, this may also be done by the planner.

The upmost tier, the task planner, is responsible for the strategic long-term planning of tasks. This layer reasons about optimal tasks from a global point of view. It deals with high level tasks and does not care much about real-

time constraints. Usually the strategic long-term goal is not changing within computation cycle-times. There can be found various implementations of the 3T model or similar tiered models for autonomous mobile robots throughout literature. Some examples can be found in [5], [6], and [7]. Whatever differences there are, all of them have a fast, reactive control layer that interacts directly with the robots hardware and provides basic behaviors.

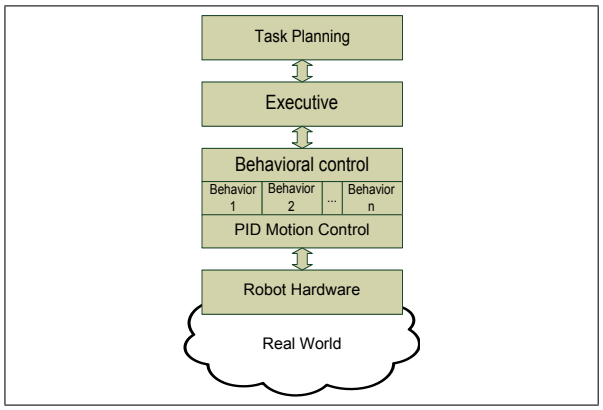


Fig. 1. Three tiered (3T) robot control architecture (author's representation derived from [5] and [6])

For students in robotics it is important to understand how this layer is working and what challenges there are. Some of these insights might be:

- What behaviors do I need for a specific task in a specific environment?
- How many behaviors should be combined?
- How can the outputs of several individual behaviors be fused in a successful way?
- What parameters can be changed in a behavioral control?
- What effect does randomization have on control?
- What are the downsides of parallel processed behaviors?
- Which difficulties are encountered at debugging?

Understanding this is crucial for students for further development of more complex robot control architectures.

Therefor a package containing a combination of a state of the art and manageable robot control software, a simulator, and compatible robot hardware has to be created. If provided, students could concentrate on getting used to behaviors and instantaneous assessment of their code within the simulation. Furthermore if the simulation shows the desired action of the robot, same code could be evaluated on real robot hardware in the real world environment. Integrating these components into a framework shall facilitate the behavioral approach to robot control a lot in education. Overcoming the need for caring about robot program structure, odometry, sensor data acquisition, or motor control, students could address their whole attention to small code-blocks for developing new behaviors and parameterizing the framework.

II. System Design

A. Mobile robot

The AmigoBot^{TM}H8 mobile robot platform was chosen to test the behavior framework and ROS environment. Because the AmigoBot is a very cost-effective differential-drive robot it perfectly serves education and collaboration projects. It comes with 8 sonar sensors and can be controlled wirelessly. Differential drive movement allows pivot turns and rotation speed of 100°/s. The swing radius is 16,5cm and the maximum forward speed is 1m/s. Two RS-232 Comm ports and three ADCs are supplied by the 44MHz z Renesas SH2-7144 microprocessor. Figure 2 shows the AmigoBots' schematic sketch. The robots' dimensions and crucial components and interfaces are displayed in this image.

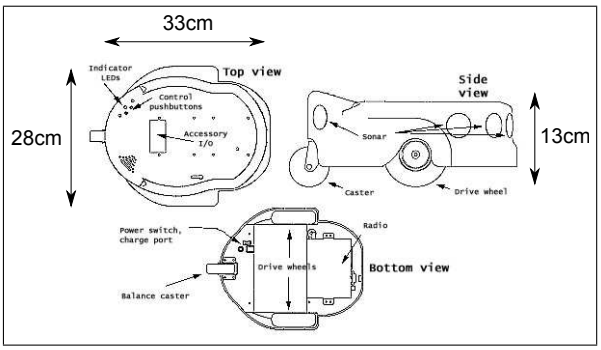


Fig. 2. AmigoBots' schematic sketch [16]

B. Middleware

Middleware is described by [9] as following:

> ... a class of software technologies designed to help manage the complexity and heterogeneity inherent in distributed systems. It is defined as a layer of software above the operating system but below the application program that provides a common programming abstraction across a distributed system.

But above that ROS can also be considered a "Meta Operating System" [10] which provides a standardized toolset and APIs. There are a couple of middleware frameworks that help to develop robotic applications. Ayssam Elkady and Tarek Sobh [11] offer a comprehensive literature survey on different robotics middleware. For this work ROS was chosen, as it supports the AmigoBot by supplying a wrapper for the "Advanced Robot Interface for Applications" (ARIA) Framework. Latter one comes with the AmigoBot and offers extensive AmigoBot functionality. Aria is a C++ library (software development toolkit or SDK) for all MobileRobots/ActivMedia platforms. It can control the robot's velocity, heading, and other parameters either through simple low-level commands or its high-level Actions infrastructure [12]. The wrapper comes as a so called "package", namely ROSARIA [14]. Furthermore ROS is a very thin layer and thus runs on embedded devices. Besides ROS is peer-to-peer based and open-source. In ROS the "name service"

fulfills the task of the required lookup mechanism. Because ROS is language neutral several programming languages can be support. ROS natively comes with C++, Python, Octave and LISP support. Through IDLs (Interface Definition Language) messages can be passed between different modules independently from the physical system. ROS follows the scheme of a microkernel design, meaning a lot of separate software components collaborate to offer a solution to a specific problem.

C. Behavior Framework

Based upon an early prototype of a previous student project the behavior framework was redesigned and embedded into ROS. It was designed to be easily reused and to be extended further by prospective students. The parameters of the behaviors can be set using text files. Also new behaviors can be added and different behaviors can be combined using a configuration file. Basically the framework consists of a sorted behavior queue. Every behavior has a priority that decides when the behavior is executed. Behaviors with higher priorities are executed before the ones with lower priority. If two or more behaviors have the same priority the mean value of their outcomes are passed to the motors. No behavior directly sets the values of the motor values, but adds up to them. The flowchart of the behavior frame work is depicted in figure 3.

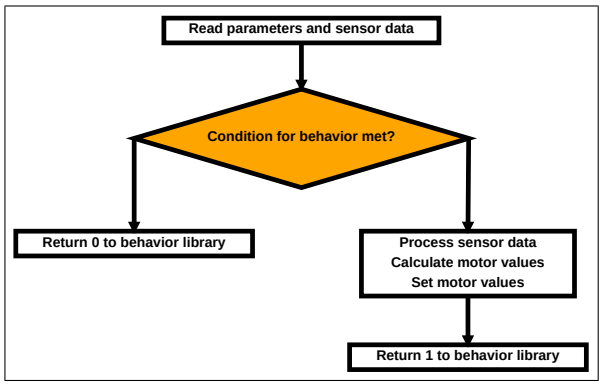


Fig. 3. Flowchart behavior framework

A very simple behavior is the bGotoXY() behavior. It first parses the robots' pose and reads the target location from the configuration file.

```
arget.setX(readParam(2,0));
target.setY(readParam(2,1));
current = robot->getPose();
```

The next step is to check, whether the robot is still out of range of the target.

```
if(current.findDistanceTo(target) >
dProximity) {
...
}
```

If this is the case then the angle to the target is calculated. Depending on whether the final destination is to the left or to the right of the robot the appropriate wheel velocities are added to the left and right wheels.

```
if(current.findAngleTo(target) >
current.getTh() + dAccuracy)
{
dVelRight -= dRotVel;
dVelLeft += dRotVel;
}
else if(current.findAngleTo(target) <
current.getTh() - dAccuracy)
{
dVelRight += dRotVel;
dVelLeft -= dRotVel;
}
...
```

If the target is just in front of the robot both wheels are set with the same value.

```
else
{
dVelRight += dMaxVel;
dVelLeft += dMaxVel;
}
```

If the robot is not within the targets' range, 1 is returned. This triggers another execution of the behavior in the next iteration. Otherwise 0 is returned. The variable dAccuracy is of importance, as the robots movement is affected by odometric errors in the real world.

D. Simulator

MobileSim [15], which also is provided by Adept MobileRobots, was used as a simulator to test the behaviors. As both, the AmigoBot and the Simulator support TCPI/IP for communication, the connection is transparent. This means that from the programmers point of view, there is no difference between the connection to the robot or the simulator. Therefore no modification had to be performed to the simulator or the users programs. It offers a 2D environment and an GUI.

E. behavior-amigo

The created ROS package "behavior-amigo" is basically a wrapper that combines the behavior framework and the ROSARIA package. It consists of the three classes AmigoBehavior, AmigoSensors and AmigoTeleoperation. The AmigoBehavior class provides all the functionalities of the behavior framework and the ROSARIA package while the latter two classes can be used to display sensor data or to remotely control the robot manually.

III. Implementation

In the course of this project the behavior-framework was embedded into ROS by creating a new package "behavior-amigo". Like the ROSARIA package its merely a wrapper for the behavior-framework. It influences the software's extensibility, modularity, reusability and compatibility.

Extensibility is increased as new robot functions can be implemented by either combining existing behaviors in a

configuration file or by adding new routines to the function 'beLib()' which selects the next behavior. Both methods don't interfere with any other parts of the software. This is very handy in the educational fields, as students really can focus on the actual robotic task instead of coding aspects.

Modularity is achieved by using ROS. Each node, in our case each program instance, is a well defined component that only communicates with other nodes using ROS topics to receive or transmit predefined messages. Further more ROS offers a tool named rxgraph that can visualize how nodes interconnect with each other in real-time. Thus very complex problems can be identified early and enventually tracked down very easily. Furthermore complex, meaning large, student projects can be split into isolated parts. These parts can be designed and tested by separate student groups and eventually be blended to create large scale applications that could otherwise not come to fruition.

Maintainability is the logic implication of the latter two features, modularity and extensibility.

Being able to add any number of existing ROS packages to this framework contributes to **reusability**.

Compatibility is also an implication of using well-defined messages between single nodes and by connecting only a fixed set of behaviors. But in this case the term compatibility exceeds the meaning of "interoperability" between different versions of the same software or other software. It also comprises the comparability of gathered empirical results.

A. Design process using framework

By using the behavior framework in conjunction with ROS the development process of new robotic application can be facilitated. The whole development process will be composed of the following sequential steps:

1) Requirement analysis
2) Software design
3) Check if the task can be achieved with an existing behavior. If so then solving the problem reduces to setting the appropriate parameters in the behavior configuration file. Otherwise proceed to the next step.
4) Check if the task can be achieved by combining existing behaviors. If so then solving the problem reduces to connecting the suitable behaviors and setting the parameters in the individual behavior configuration files. Otherwise proceed to the next step. If one could solve the problem at this point the implementation is complete.
5) Check if the problem can be solved by incorporating an existing ROS package. This can be the case if functionality beyond the robot behavior is required, e.g accessing new hardware. If this is sufficient then the implementation is completed. If there is no package then one has to proceed to the next step.
6) Check if either a new behavior or a new software functionality. e.g accessing new hardware, is necessary. Either wise the desired functionality has to be coded.

One can see that the necessity to actually code can be circumvented in most cases. This allows students to focus on actual robotic tasks. The previously described process is depicted in figure 4. Blue process boxes comprise coding work, whereas grey process boxes require either only manipulation of behavior configuration files or plugins of ROS packages. The white processes only represent conceptual work, including the requirement analysis and software design. Only the latter phase is influenced indirectly by the framework.

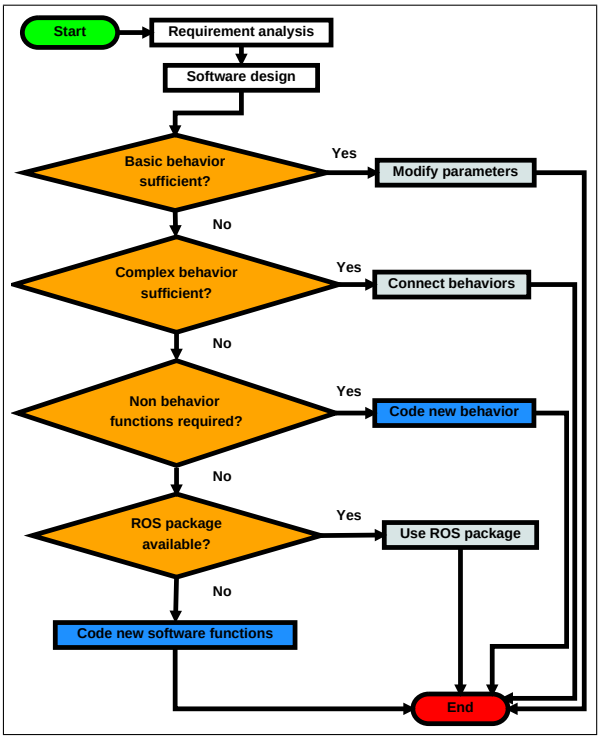


Fig. 4. Flowchart of design process using framework

B. Design process without framework

1) Requirement analysis
2) Software design
3) Check if there any libraries that supply the desired functionality. This can either happen on the internet or university intranet. If there aren't any libraries then one has to implement them by writing code. Otherwise one has to proceed to the next step.
4) Now one has to verify if the libraries are compatible. Meaning that they are available for the desired platform. Either way they have to be integrated or ported to fit the project.

The previously described process is depicted in figure 5.

C. Comparison

Figures 4 and 5 make it obvious, that without the framework the necessity to write programm code can never be avoided. In figure 4 one can identify three out of five possible paths that don't require coding at all.

D. Example implementation process

Now an example will be given by showing an actual implementation using the framework. In this example the

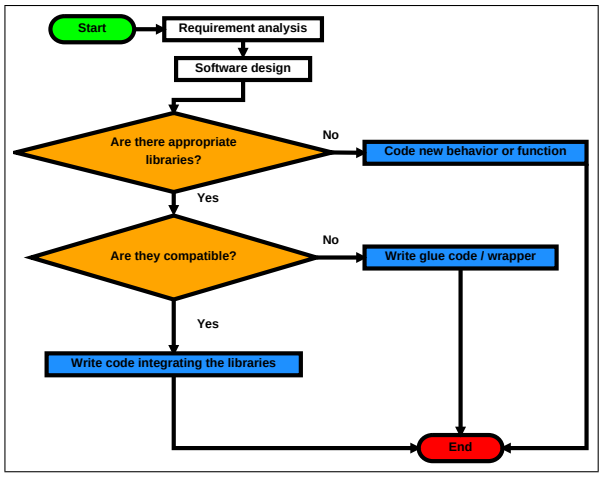


Fig. 5. Flowchart of design process without framework

robot's task is to reach a certain point without collisions. Two behaviors are supplied for this purpose, namley a behavior called 'bSimpleAvoid()' and one named 'bGotoXY()'. The second behavior navigates the robot to a given point in the global reference system. After the design phase one has to identify the appropriate behaviors and adapt their configuration files. The structure of a file is very simple and looks like in the listing below.

```
Destination X:6000
Destination Y:-5000
Maximum Speed:200
Proximity:50
Rotational Speed:10
Accuracy:1
Mean:1
Sigma:0
```

The mean and the sigma value allow to blend some randomnes into the calculated values. This can be helpful if the robot got stuck in a deadlock. Such situations can occur, in case of a sensor toggling between collision and a distance above the trigger distance. Taking a turn at a corner is an example for such a case.

Different behaviors are then combined by putting them into the behaviors' queue configuration file together with their priority. The listing below describes how this is done for this example.

```
2:5
1:5
```

Every behavior has an ID, which is identical to the configurations file name. In this case the IDs are 1 and 2. Both behaviors have the same priority of five. This means that both contribute to the motor value. So implementing an individual behavioral control layer only requires the textual manipulation of three textfiles. Figure 6 shows the structure of the application shown by the ROS software rxgraph. Each node is an instance of an application that communicates by the means of ROS topics. There is no difference if these nodes run on the same physical machine or are distributed in a network. Rxgraph allows real-time information on the topic values and the status of the nodes. If communication fails it can easly be pinpointed. If one would not use the ROS enviroment debugging communication related issues become very hard as one would have to crawl logfiles or network dumps. The picture also shows that the individual nodes are encapsulated and only communicate through predefined messages. In this case the topics are /amigo_behavior/sonar, /amigo_behavior/pose and /rosout.

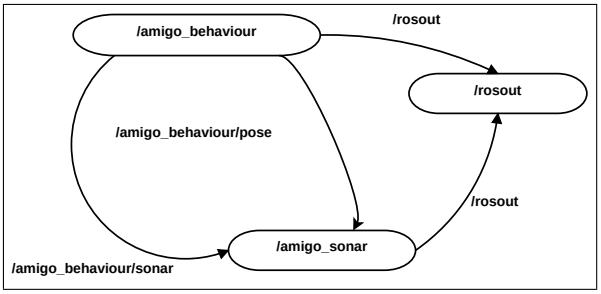


Fig. 6. ROS strucutre of application

IV. Results

The result of this work is a framework that allows students a very structured but still flexible implementation process of robotic applications. This is essential for the students learning success. Figure 7 shows the simulations' result when blendig both behaviors. With very litte effort a quite complex behavior can be implemented. Summing up the framework comprises the following components and thus offering a complete RDE:

- A "Meta Operating System" namley ROS.
- Robot behaviors that can easily be modified and connected.
- The ROS package "behavior-amigo" that embeds the behavior framework into ROS.
- The simulator "MobileSim" that transparently connects to the software.
- Robotic hardware that is supported by the framework. Compatible hardware comprises robots supplied by Adept MobileRobots and especially the AmigoBot.

Another test has been conducted using a simple test environment. The result of the simulation and the real robots' behavior were then compared. Figure 8 shows the outcome of the test. The plot was created by using the "behavior-amigo" packages' logging features. From the programs point of view there was no difference between both test runs. The colored arrows indicate the sensor's values.

In the top plot the data of the simulation is displayed. An obstacle to the left of the robot is detected by the sensors. This circumstance is represented by shorter sonar vector values. In contrast to the smooth and ordered outcome of the simulation, the real world results are less accurate. Despite of this, the simple behavior accomplishes the robots' mission.

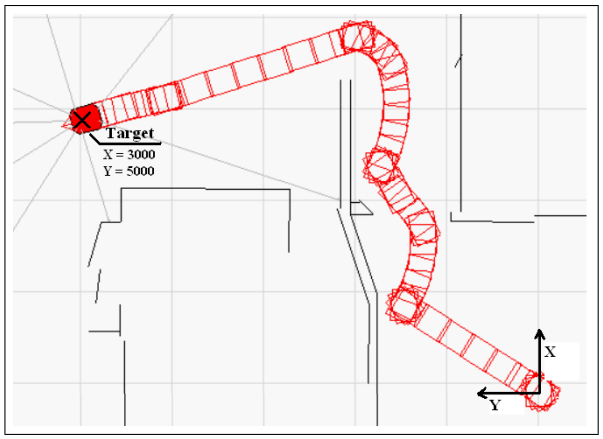


Fig. 7. Simulation result of both behaviors combined

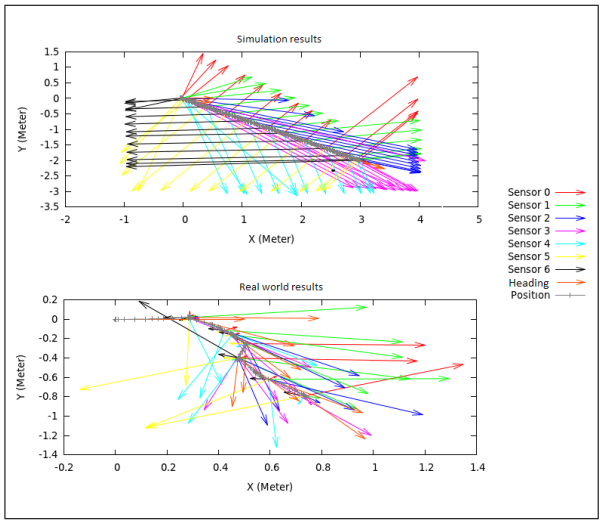


Fig. 8. Comparison simulation and real world behavior

V. Future work

Until now the framework hasn't been inserted into any curriculum. However the Departments of Mechatronics is currently working on providing a course, which includes several tasks that can be solved with the framework, for the upcoming term. Once the framework is in use, other projects that investigate the benefit of the framework would be of interest. The students' feedback could then be incorporated into future versions of the framework.

A graphical user interface would also be a desirable component to increase the usability of the framework. A graphical behavior design program would help to visualize complex interactions between behaviors. It would have to be able to parse the configuration files and then visualize the behaviors interactions in a state machine like representation. Alternatively one should be able to connect different behaviors in a drag and drop like fashion.

Real time feedback on the different behavior's activities would be beneficial for debugging purposes. Furthermore, extending the available behaviors pool would be another goal for future work.

VI. Conclusion

Because of the increasing complexity of software development in the field of robotics "Robotic Development Environments (RDE)" have gained more and more importance [8]. The complexity of the implementation process could be reduced dramatically. This is essential, so that students can really focus on robotic related problems, such as navigation and mapping. Supplying students with ready to use frameworks goes with this design paradigm.

Acknowledgment

This work was supported by "Stadt Wien Kompetenzteam für Mobile Roboter" at UAS Technikum-Wien. According to the aim of this grant program, bringing together research and education, the outcome of this work will be used at the bachelor program for mechatronics at UAS Technikum Wien.

References

[1] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hhnel, C. Rosenberg, J. Schulte, and D. Schulz,"Probabilistic algorithms and the interactive museum tour-guide robot Minerva," Int. J. of Robotics Research (IJRR) 19(11):972-999, 2000.

[2] Y. Hada, K. Takase, H. Gakuhari, and E. I. Hemeldan, "Delivery service robot using distributed acquisition, actuators and intelligence," in Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004, pp. 2997-3002.

[3]] F. Sandt and L.-H. Pampagnin, "Perception for a transport robot in public environment," in Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'97, vol. 1, Sep 1997, pp. 360-365.

[4] Swisslog, "Integrated logistics solutions for warehouses, distribution centers and hospitals [online]," Available from http://www.swisslog.com/ [accessed 13 March 2013], 2013.

[5] R. Siegwart, I. R. Nourbakhsh, and D. Scaramzza, Autonomous Mobile Robots. MIT Press, 2nd ed, 2011

[6] B. Siciliano et al., Handbook of Robotics. Springer, 2008

[7] Arkin, R.C., Behavior-Based Robotics. Cambridge MA, MIT Press, 1998

[8] J. Kramer and M. Scheutz, "Development environments for autonomous mobile robots: A survey," in *Autonomous Robots*, vol. 22, pp. 101–132.

[9] D. E. Bakken and M. Api, "Middleware," 2001.

[10] S. Rainwater. [Online]. (08.03.2013). Available: http://www.willowgarage.com/pages/software/ros-platform

[11] A. Elkady and T. Sobh, "Robotics middleware: A comprehensive literature survey and attribute-based bibliography," *Journal of Robotics*, vol. 2012, no. 5, p. 115, 2012.

[12] Aria - mobilerobots research and academic customer support. (2013). [Online]. Available: http://robots.mobilerobots.com/wiki/ARIA

[13] Ros — willow garage. [Online]. (08.03.2013). Available: http://www.willowgarage.com/pages/software/ros-platform

[14] Rosaria - ros wiki. [Online]. (12.03.2013). Available: http://www.ros.org/wiki/ROSARIA

[15] Mobilesim robot simulators. (12.03.2013). [Online]. Available: http://www.mobilerobots.com/software/mobilesim.aspx

[16] Team AmigoBot Operations Manual, 4th ed., MobileRobots Inc., Amherst, NH, 2007, pp. 14

Design, Modeling and Control of a Self-Balancing Two-Wheeled Vehicle

F. Johannes Kilian[1], Hubert Gattringer[1], Klemens Springer[1] and Hartmut Bremer[1]

***Abstract*— The following contribution introduces the design, modeling and control of a two-wheeled self-balancing vehicle. This mobile robot is about 60cm tall and uses two wheels actuated by two DC gear drives to execute transport tasks. An embedded board and a microcontroller enable the control of the wheels and assume the path planning. Several sensors, such as a Microsoft Kinect as well as a gravitation and an angular rate sensor, stabilize the robot and make a communication between robot and humans possible.**

Due to the nonholonomic constraints of the wheels, only few modeling methods are feasible. The obtained model serves as basis for the following control design. A flatnessbased approach provides an excellent possibility to follow a trajectory, but requires an underlaid velocity controller and continuous curvature paths. The combination of the well designed trajectory controller and the applied paths, which are composed by straight lines, circles and clothoids, enable a good performance in simulation and experiment.

I. INTRODUCTION

The two-wheeled, self-balancing Segway PT found one's way into everyday life of human transport. Dean Kamen introduced this vehicle in 2001 and therefore changed the focus of personal transport. The Institute of Robotics of the Johannes Kepler University Linz researches for several years on the design and control of small two-wheeled, self-balancing, Segway-similar mobile robots, see [1], [2] and [3] for further details. The main tasks of this research project is the advancement of autonomous, mobile robots with two wheels. Compared with conventional three- or four-wheeled robots, this self-balancing two wheeled robot requires only two drives and resolves transport tasks in comparative good way. Beyond the motion control challenges, all other modules of an autonomous robot, such as localization, path planning and human-machine-interface, only may draw small spatial and computing resources. The considered robot is shown in Fig. 1. It is about $60\,\mathrm{cm}$ tall, $25\,\mathrm{cm}$ wide and has a weight of $4.6\,\mathrm{kg}$.

The paper is organized as follows. In section 2, we give an overview of the mechanical and electrical design of the robot. Section 3 introduces a kinematic and dynamic model, which is used for the control design in section 4. A stabilization and a trajectory controller demonstrate the potential of this mobile robot for transport tasks. Section 5 deals with the path planning and passes to the experimental results in section 6, which concludes this contribution.

[1]Institute for Robotics, Johannes Kepler University Linz, Altenbergerstr. 69, 4040 Linz, Austria.

Fig. 1. The two-wheeled, self-balancing mobile robot in front of the newly built Science Park of the Johannes Kepler University Linz

II. DESIGN

The contemplated robot consists of three bodies: the two wheels and the corpus. The corpus includes the battery and the Microsoft Kinect in the top and the electronic box at the bottom. This box contains two $20\,\mathrm{W}$ Maxon motors connected to planetary gears transmitting the motor speed by a gear ratio of 27. Furthermore, all sensors and electronic devices can be found in the electronic box.

The electrical design of the robot is shown in Fig. 2. The computational heart of the controller is an embedded computer board (ADVANTECH PCM-9362) including a $1.67\,\mathrm{GHz}$ processor. This board exercises the control algorithms and the interface processing. An Infineon C167 microcontroller communicates with the motor amplifier and evaluates the sensors - an accelerometer sensor as well as an angular rate sensor and two encoders. Furthermore, it receives and delivers the signals to the embedded board via

a self-defined protocol using a parallel port interface. The embedded board communicates with several human-machine interfaces (HMIs) via USB or Wifi. The Microsoft Kinect and Android applications on any compatible hardware provide a possibility to control the behavior of the mobile robot.

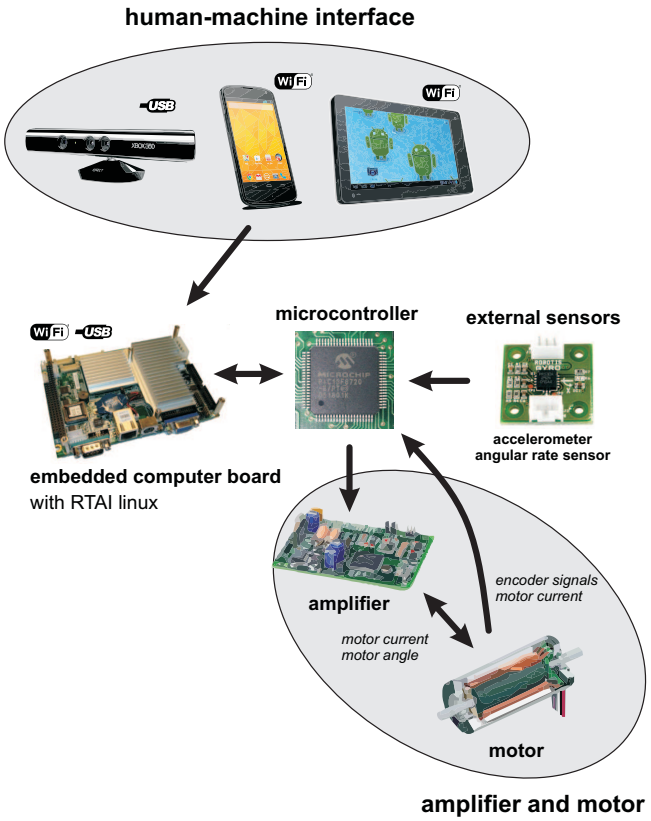


Fig. 2. The electric design of the two-wheeled, self-balancing mobile robot

The upward angle of the corpus is measured by means of complementary filtering, see [4]. Measuring the accelerations in two directions provides a good estimation for the angle in terms of a statical process. The dynamic effects (high frequencies) are taken into account by using angular rate measurements, see [2] and [5]. Because the microcontroller is aware of all necessary information, an implemented LQR controller stabilizes the mobile robot, if the embedded board is not active. Therefore, stabilization and resulting intactness of the robot can be guaranteed at any time.

III. KINEMATIC AND DYNAMIC MODELING

It is in the nature of things that conventional wheels are unable to move lateral to the front direction. This fact leads to the resulting condition

$$v_Q = 0, \tag{1}$$

see Fig. 4. Due to this nonholonomic constraint, only those modeling methods can be considered which take these special constraints into account. Hence, for instance the Lagrange II formalism leads to incorrect dynamic equations. Common modeling methods, such as the Maggi equation or the Hamel-Boltzmann equations, insert the nonholonomic constraint only in the last step and therefore require the modeling with a higher dimension as actual necessary. The Projection Equation

$$\sum_{i=1}^{N} \begin{bmatrix} \dfrac{\partial_R \mathbf{v}_c}{\partial \dot{\mathbf{s}}} \\ \dfrac{\partial_R \boldsymbol{\omega}_c}{\partial \dot{\mathbf{s}}} \end{bmatrix}_i^T \begin{bmatrix} \left({}_R\dot{\mathbf{p}} + {}_R\widetilde{\boldsymbol{\omega}}_{IR}\ {}_R\mathbf{p} - {}_R\mathbf{f}^e\right)_i \\ \left({}_R\dot{\mathbf{L}} + {}_R\widetilde{\boldsymbol{\omega}}_{IR}\ {}_R\mathbf{L} - {}_R\mathbf{M}^e\right)_i \end{bmatrix} = 0 \tag{2}$$

has proved itself to be an excellent approach to deriving the equations of motion for multibody, nonholonomic systems and minimizes simultaneously the dimensions of the minimal velocities, see [6] for details. This method projects the linear ($\mathbf{p} = m\,\mathbf{v}_c$) and angular ($\mathbf{L} = \mathbf{J}\,\boldsymbol{\omega}_c$) momenta of the ith body in the space of the minimal velocities $\dot{\mathbf{s}}$, see [1] for a comparative study of modeling methods for such a robot. The choice of the minimal velocities

$$\dot{\mathbf{s}}^T = \begin{pmatrix} \dot{\alpha} - \dot{\theta} & \dot{\beta} - \dot{\theta} & \dot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{\mathbf{s}}_1^T & \dot{s}_2 \end{pmatrix}, \tag{3}$$

and the minimal coordinates

$$\mathbf{q}^T = \begin{pmatrix} x & y & \gamma & \theta \end{pmatrix}, \tag{4}$$

using the wheel angles α and β and the inclination angle θ, implies already the nonholonomic contstraint $v_Q = 0$. The in general nonlinear kinematical relationship between the minimal coordinates $\mathbf{q}$ and the minimal velocities $\dot{\mathbf{s}}$ is given by

$$\dot{\mathbf{s}} = \mathbf{H}(\mathbf{q})\,\dot{\mathbf{q}}. \tag{5}$$

Carrying out the Projection Equation, we obtain the equations of motion

$$\mathbf{M}(\mathbf{q})\,\ddot{\mathbf{s}} + \mathbf{g}(\mathbf{q}, \dot{\mathbf{s}}) - \mathbf{B}\,\mathbf{u} = 0, \tag{6}$$

which serve as basis for the following control design.

IV. CONTROL

The control of the self-stabilizing mobile robot is split into two different tasks: on the one hand, the stabilization of the inclination angle is the most important challenge. An overlaid trajectory controller seems to be indispensable for transport tasks on the other hand. Both controllers are discussed in detail within the following section.

A. Stabilization and Velocity Controller

In order to safe the mobile robot from damage, the stabilization of the inclination angle is absolutely necessary. Therefore, a partial feedback linearization gives the possibility to combine the stabilization with a velocity controller, which deals as basis for the following trajectory control.
The structure of the equations of motion

$$\begin{aligned} \mathbf{M}_{11}\,\ddot{\mathbf{s}}_1 + \mathbf{M}_{12}\,\ddot{s}_2 + \mathbf{h}_1 &= \mathbf{B}_1\,\mathbf{u} \\ \mathbf{M}_{21}\,\ddot{\mathbf{s}}_1 + M_{22}\,\ddot{s}_2 + h_2 &= 0, \end{aligned} \tag{7}$$

using the motor torques $\mathbf{u}^T = \begin{pmatrix} M_\alpha & M_\beta \end{pmatrix}$, allows the linearization by using the new system input $\mathbf{v}$

$$\mathbf{u} = \mathbf{B}_1^{-1}\left(\overline{\mathbf{M}}_{11}\,\mathbf{v} + h_2\right) \tag{8}$$

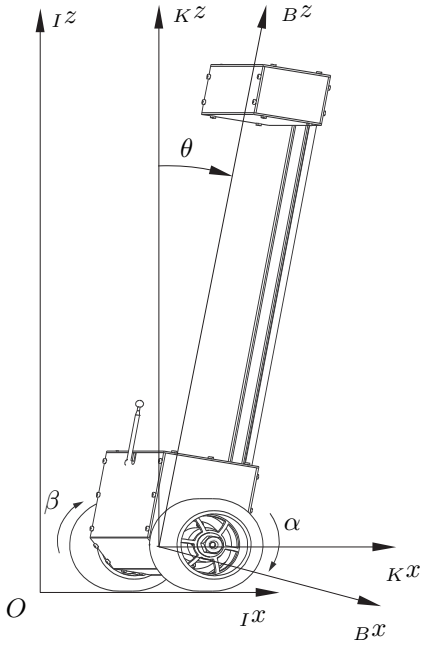


Fig. 3. Sketch of the considered mobile robot (front view)

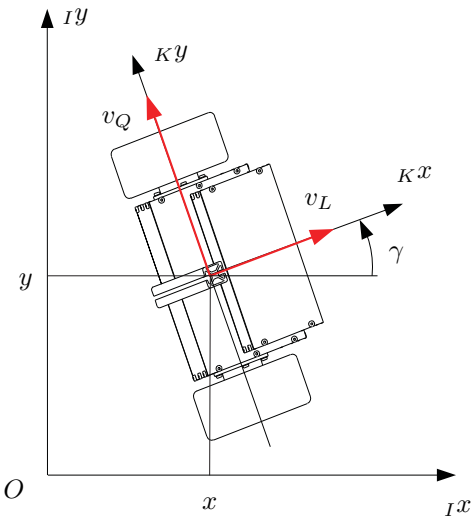


Fig. 4. Sketch of the considered mobile robot (horizontal view)

with the abbreviations

$$\begin{aligned} \overline{\mathbf{M}}_{11} &= \mathbf{M}_{11} - \mathbf{M}_{12}\, M_{22}^{-1}\, \mathbf{M}_{21} \\ \overline{\mathbf{h}}_1 &= \mathbf{h}_1 - \mathbf{M}_{12}\, M_{22}^{-1}\, h_2. \end{aligned} \tag{9}$$

The input transformation of Eq. 8 leads to a partially linearized system

$$\begin{aligned} \ddot{\mathbf{s}}_1 &= \mathbf{v} \\ \ddot{s}_2 &= -M_{22}^{-1}\left(\mathbf{M}_{21}\,\mathbf{v} + h_2\right), \end{aligned} \tag{10}$$

which can be linearized around the inclination angle $\theta = 0$. As a result, a LQR controller can be introduced based on the minimization of the cost function

$$J = \sum_{k=1}^{\infty} \left(\mathbf{x}_k^T \mathbf{Q}\, \mathbf{x}_k + \mathbf{v}_k^T \mathbf{R}\, \mathbf{v}_k\right), \tag{11}$$

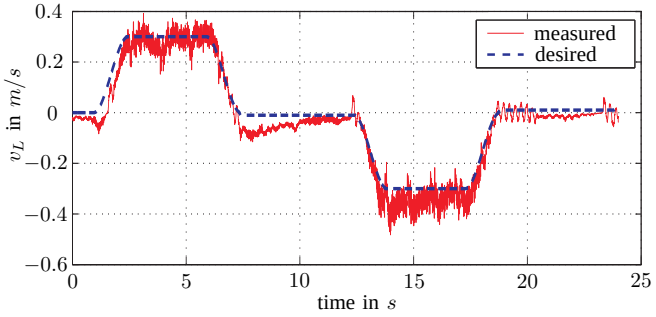


Fig. 5. Experimental results of the velocity control with the calculation of the desired inclination angle

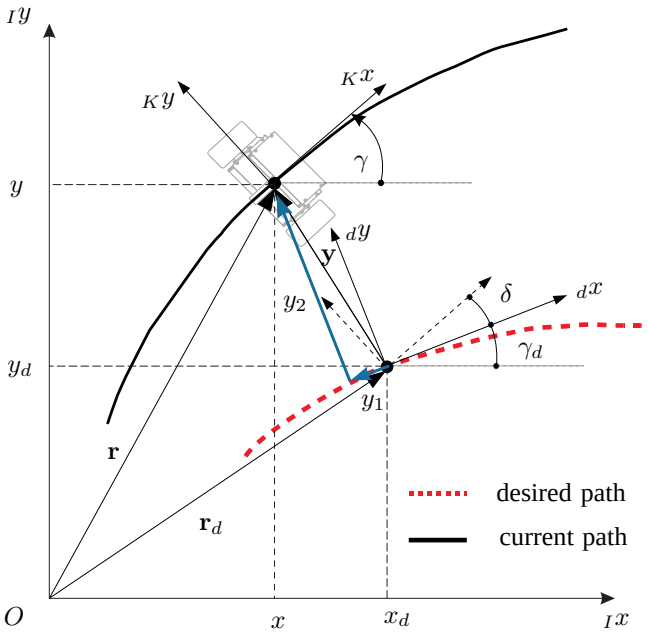


Fig. 6. Kinematical model of the vehicle along a trajectory

using the system states $\mathbf{x}^T = \begin{pmatrix} \mathbf{q}^T & \dot{\mathbf{s}}^T \end{pmatrix}$ and the input $\mathbf{v}$. The LQR control design yields the state feedback control

$$\mathbf{v} = -\mathbf{K}\,\mathbf{x}. \tag{12}$$

The desired minimal coordinates $\mathbf{q}$ and minimal velocities $\dot{\mathbf{s}}$ result from the kinematic relationships between the motor angles and the longitudinal and rotational velocity.
In order to increase the maximum speed of the mobile robot, an acceleration-proportional inclination angle θ deals as controller input. Therefore, a linearization of Eq. 6 along the desired longitudinal acceleration $\dot{v}_{L,d}$ enables an explicit calculation

$$\theta_d = \theta_0 + h\left(\mathbf{q}, \dot{\mathbf{s}}\right)\, \dot{v}_{L,d} \tag{13}$$

of the desired inclination angle. Experimental results of the partial feedback linearization are shown in Fig. 5 and demonstrate the high coincidence between the desired and measured robot speed.

B. Trajectory Controller

Based on the established velocity controller, the trajectory tracking control relies on the kinematic model (see Fig. 6)

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\gamma} \end{pmatrix} = \begin{pmatrix} v_L \cos\gamma \\ v_L \sin\gamma \\ \dot{\gamma} \end{pmatrix} \tag{14}$$

of the mobile robot and uses the velocities

$$\boldsymbol{\eta} = \begin{pmatrix} v_L \\ \dot{\gamma} \end{pmatrix} = \begin{pmatrix} \eta_1' \\ \eta_2' \end{pmatrix} \dot{\sigma} \tag{15}$$

as input. The variable σ denotes the path coordinate and $'$ the spatial derivative $\partial / \partial\sigma$. Rudolph [7] and Woernle [8] show that the tracking error $\mathbf{y}^T = \begin{pmatrix} y_1 & y_2 \end{pmatrix}$, represented in the desired local frame of the path $({}_dx,\ {}_dy)$, emerges as flat output. Hence, all states and inputs of the kinematic model can be expressed as a function of the outputs y_1, y_2 and their spatial derivatives. Therefore, the position of the mobile robot can be expressed as

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_d \\ y_d \end{pmatrix} + \underbrace{\begin{bmatrix} \cos\gamma_d & \sin\gamma_d \\ -\sin\gamma_d & \cos\gamma_d \end{bmatrix}}_{\mathbf{A}_{Id}} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \tag{16}$$

whereas the orientation

$$\gamma = \gamma_d + \delta = \gamma_d + \arctan\left(\frac{y_2' - c_2(x,y,\sigma)}{y_1' - c_1(x,y,\sigma)}\right) \tag{17}$$

requires the spatial derivatives of the flat output

$$\begin{aligned} \mathbf{y}' &= \mathbf{A}_{Id}'^T \begin{pmatrix} x - x_d \\ y - y_d \end{pmatrix} + \mathbf{A}_{Id}^T \begin{pmatrix} \eta_1' \cos\gamma - x_d' \\ \eta_1' \sin\gamma - y_d' \end{pmatrix} \\ \mathbf{y}' &= \begin{pmatrix} c_1(x,y,\sigma) \\ c_2(x,y,\sigma) \end{pmatrix} + \begin{pmatrix} \cos\delta \\ \sin\delta \end{pmatrix} \eta_1', \end{aligned} \tag{18}$$

see [5] and [8] for more details. The flat representation of the system input emerges from Eq. 18 and results in

$$\begin{aligned} \eta_1' &= \frac{y_1' - c_1(x,y,\sigma)}{\cos\delta} \\ \eta_2' &= \gamma_d' + \delta' = \gamma_d' + \frac{(y_2'' - c_2')\cos\delta - (y_1'' - c_1')\sin\delta}{\eta_1'}, \end{aligned} \tag{19}$$

see [8] for details. By eliminating the highest spatial derivative by a new system input

$$\begin{aligned} y_1' &\rightarrow w_1 \\ y_2'' &\rightarrow w_2, \end{aligned} \tag{20}$$

asymptotic stability by choosing an additional error feedback

$$\begin{aligned} w_1 &= y_{1,d}' + a_0(y_{1,d} - y_1) \\ w_2 &= y_{2,d}'' + b_1\left(y_{2,d}' - y_2'\right) + b_0(y_{2,d} - y_2) \end{aligned} \tag{21}$$

is obtained. Compared to conventional control algorithms, this feedback method offers its error dynamics in the local reference frame. This approach distinguishes itself due to the nonholonomic constraint of the two-wheeled, mobile robot and therefore enables much higher velocities. In comparison with a control algorithm using the initial frame for the error dynamics, the velocity v_L can be increased from $0.1\,\mathrm{m/s}$ up to $0.4\,\mathrm{m/s}$. Regarding the derivative γ_d' in the feedback algorithm (see Eqns. 19 and 21), the desired trajectory requires continuous curvature paths. Therefore the geometric path planning must be discussed in detail, see section V.

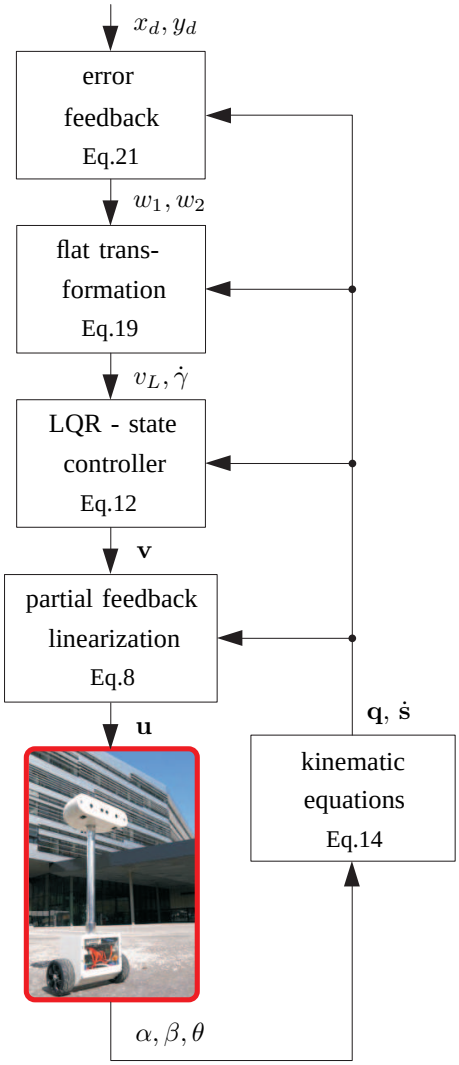


Fig. 7. Sketch of the overall trajectory control scheme.

C. Overall control scheme

To sum up the trajectory control, Fig. 7 shows the overall control scheme of the trajectory controller. The desired path deals as input for the error feedback of Eq. 21. These results serve as input again for the flat transformation and further on for the velocity control, see Eq. 8. The system states and velocities are calculated by the kinematic equations using the measurement of α, β and θ, see Eq. 14. Apparent from Eq. 19, the desired path (x_d, y_d) has to satisfy several conditions and is therefore discussed in the following section.

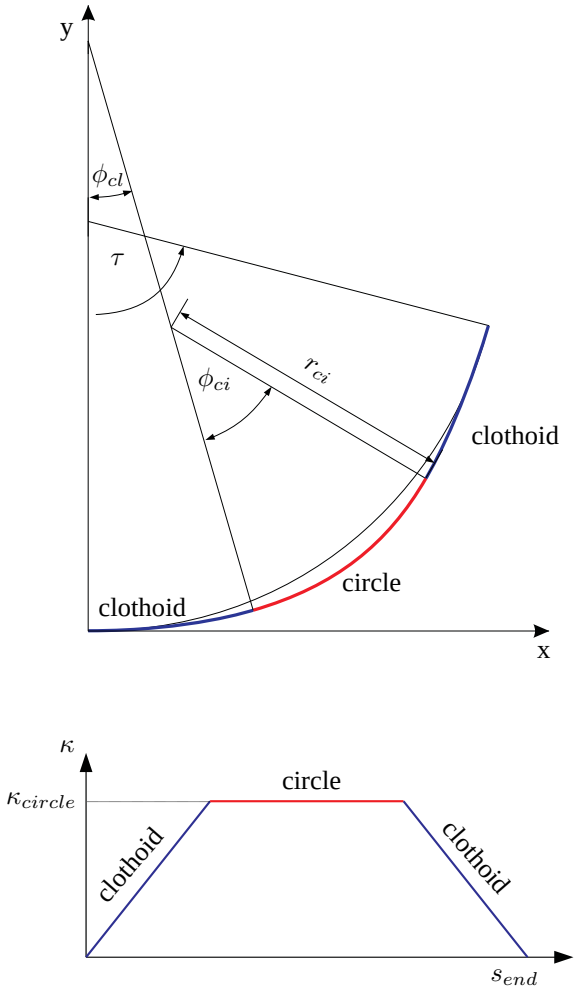


Fig. 8. Sketch of a circle-clothoid combination and its continuous curvature path.

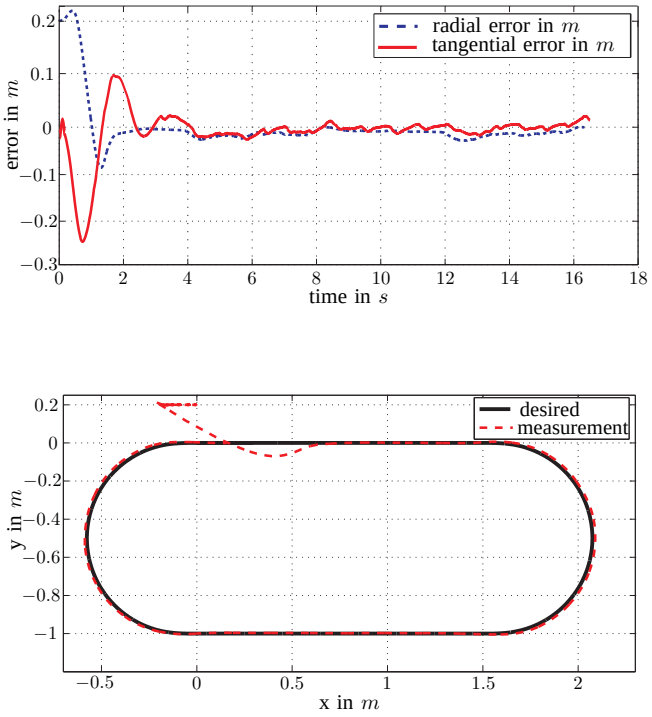


Fig. 9. Measurement results along a desired path with an initial error

V. PATH PLANNING

The geometric path of the desired trajectory is split into the basic elements *line* and *circle*. Using combinations of these, almost all arbitrary paths can be generated. However, the assembly of these two elements causes a discontinuous curvature due to the constant curvature of the *circle*. Therefore, clothoids (resp. Euler spirals), which are characterized by a continuous curvature, are required to obtain a smoother run of the geometrical path and enable the control law of Eqns. 19 and 21.

As shown in Fig. 8, the curvature of the clothoid changes linear with the arc length. Hence, the constant curvature of the circle can be reached continuously. For this reason a circle-element of the path consists of the assembly clothoid-circle-clothoid. For more information on path planning using clothoids, please refer to [9], [10] and [11].

VI. MEASUREMENTS AND CONCLUSIONS

The flatnessbased trajectory controller as well as the continuous curvature path planning are implemented on the embedded board of the mobile robot (Fig. 1). Figure 9 shows measurement results of the vehicle tracking a desired path with an initial error of $200\,\mathrm{mm}$ in radial direction. After the compensation of the initial error, the vehicle follows the desired trajectory in a very accurate way and achieves a radial error less than $10\,\mathrm{mm}$. The combination of the partial feedback linearization with the aid of the velocity control and the trajectory controller enables a maximum longitudinal speed for this specific trajectory of $0.4\,\mathrm{m/s}$. Fortunately, the trajectory controller uses an error dynamics in the local reference frame and allows therefore high velocities and precision during trajectory tracking.

By means of this good experimental results, this control concept is suitable for many applications in the field of mobile robotics. Following works will intensify the integration of HMIs and their tasks in terms of gesture recognition, voice control and path planning.

ACKNOWLEDGMENT

The authors gratefully acknowledge support of the peer-reviewed Austrian Center of Competence in Mechatronics (ACCM) for the present work.

REFERENCES

[1] W. Höbarth, H. Gattringer and H. Bremer, Methodenvergleich bei der Modellierung eines Mini Segway, PAMM Special Issue: 77th Annual Meeting of the International Association of Applied Mathematics and Mechanics (GAMM), WILEY-VCH Verlag, Berlin, 2006, pp. 95-96.

[2] H. Gattringer, A. Rohrhofer and W. Höbarth, Modelling and Control of a Mini Segway, Proceedings of 5th International Conference on Computational Intelligence, Robotics and Autonomous Systems, CIRAS 2008, Linz, 2008, pp. 199-203.

[3] F. J. Kilian, H. Gattringer and H. Bremer, Modeling and Quasi-Static Trajectory Control of a Self-Balancing Two-Wheeled Vehicle, PAMM Special Issue: 83rd Annual Meeting of the International Association of Applied Mathematics and Mechanics (GAMM), WILEY-VCH Verlag, Darmstadt, 2012.
[4] A.J. Baerveldt and R. Klang, A Low-cost and Low-weight Attitude Estimation System for an Autonomous Helicopter, Intelligent Engineering Systems, Hungary, 1997, pp. 391-395.
[5] H. Gattringer, Starr-elastische Robotersysteme - Theorie und Anwendung, Springer, 2011.
[6] H. Bremer, Elastic Multibody Dynamics - A Direct Ritz Approach, Springer Science+Business Media, 2008.
[7] J. Rudolph, Flachheitsbasierte Folgeregelung, Lecture Notes Johannes Kepler University Linz, Linz, 2003.
[8] C. Woernle, Flatness-Based Control of a Nonholonomic Mobile Platform, Zeitschrift für Angewandte Mathematik und Mechanik, Vol. 78 Suppl. 1, 1998, pp. 43-46.
[9] A. Scheuer, T. Fraichard, Continuous-Curvature Path-Planning for Car-Like Vehicles, IEEE-RSJ International Conference on Intelligent Robots and Systems, Vol. 2, Grenoble, 1997, pp. 997-1003.
[10] A. Scheuer, T. Fraichard, From Reeds and Shepp's to Continuous-Curvature Paths, IEEE Internation Conference on Advanced Robotics, Tokyo, 1999, pp. 585-590.
[11] B. Müller, J. Deutscher, S. Grodde, Continuous Curvature Trajectory Design and Feedforward Control of Parking a Car, IEEE Transactions on Control Systems Technology, Vol. 15(3), 2007, pp. 541-553.

In-pipe Cleaning Mechanical System for DeWaLoP Robot - Developing Water Loss Prevention

Luis A. Mateos and Markus Vincze

Abstract— **After more than 50 years the connections between fresh water pipes (800-1000mm diameter) need to be repaired due to aging and dissolution of the filling material. Only in Vienna 3000km of pipes need to be improved, which requires a robotic solution. This paper describes the in-pipe cleaning system used by the DeWaLoP robot, its configurations, mechanical properties, kinematics and preliminary cleaning test results. This new approach for in-pipe cleaning mechanism, mimicking a double cylindrical robot. With an end effector, such as a power tool (grinder) mounted on one of the arms, while a drive wheel is located on the second arm, enabling rotation to the entire cleaning system with high resolution steps. Additionaly, our approach prevents the cleaning mechanism from damaging the pipe, reacting to overcome any excessive force. To do so, its mechanical arrangement includes a suspension system on the double cylindrical arm to react similar as the arms from a human operator when working with a cleaning tool over corroded surfaces, enabling the tool to retract if vibrates or jumps back, instead of passing all these noxious forces to the pipe.**

I. Introduction

Fresh water pipelines are prone to damages due to aging, excessive traffic and geological changes. Resulting from these damages, the pipe-joints may not be completely hermetic and water loss along the pipeline may occur. Leakage is not only a problem in terms of wasting an important resource, it also results in an economic loss in form of damages to the supplying system and to foundations of roads and buildings too [1] [2].

The installation or replacement of pipelines implicates high cost and use of heavy machinery, such as cranes. In addition, side effects may occur, such as constructions sites placed along streets, blocking pedestrian and traffic tracks [3]. The size of pipes transporting water between residential areas and industrial parks is normally ranged from 800*mm* to 1200*mm* in diameter, which make it possible for one man to enter. Consequently, human operators can access the pipe and attempt to clean and repair it, as shown in figure 1. Nevertheless, this creates a special situation that presents safety and health risk to the human operator [4].

Currently, the applications of robots for the maintenance of the pipeline utilities are considered as one of the most attractive solutions available. Nevertheless, to substitute skilled human operators, pipe redevelopment requires mechanisms with high degree of mobility, able to move along the pipeline, overcoming obstacles, extreme environments, and with high accuracy clean and repair specific areas of the pipe [5] [6] [7].

Pipe cleaning methods, such as: water pressure and cleaning by friction from rotating flails are commonly used among in-pipe robots. For DeWaLoP project, the water pressure methods is not the best option. Since the required cleaning must take into account not to damage the pipe-joint hemp pack, caulked up with a lead ring in the 1920's. Thus, the only available cleaning method is by friction with wire brushes disks and grinding heads to remove the corrosion from the joint socket.

This paper presents related work from in-pipe cleaning mechanism developed by academia and industry and compare them to our proposed cleaning mechanism. Following the state-of-the-art, an overview of the DeWaLoP in-pipe robot is given, to explain its overall design and functionality. Then, a detailed description of the developed cleaning mechanism is presented, its configurations, mechanical properties, kinematics and preliminary test results.

II. Related Work

In-pipe cleaning robots can be categorized to two types: 1) Pressure-based cleaning robots and 2) Tool-based cleaning robots. In this section, we present the typical in-pipe cleaning robots developed both from academia and industry.

A. Pressure-based cleaning methods

J. Saenz [8] presents a cleaning system able to work efficiently and control the pressure of the nozzle through a relative accurate positioning to the pipe wall. They commented "A common risk when cleaning with high pressure water is the possible damage to the surface from overly applied pressure. This risk can be minimized with such a cleaning system where the cleaning parameters can be carefully controlled and monitored". Even if the pressure can be controlled, for cleaning pipe-joint this is not recommended, due to the pressure exerted by the water, pushing the hermetic seal of the pipe-joint.

B. Tool-based cleaning - Impact abrasion methods

1) GRISLEE - Gasmain Repair and Inspection System for Live Entry Environments: The GRISLEE is designed to be modular, so different kinds of in situ repairs are possible. The cleaning system consists of flails, which expand when rotates and cleans the surface by impact abrasion method. The system has a compact size, and is able to work in different pipe sizes [9].

Luis A. Mateos and Markus Vincze are with Automation and Control Institute (ACIN), Vienna University of Technology (TU WIEN), Gusshausstrasse 27 - 29 / E376, A - 1040, Austria. {mateos,vincze} at acin.tuwien.ac.at

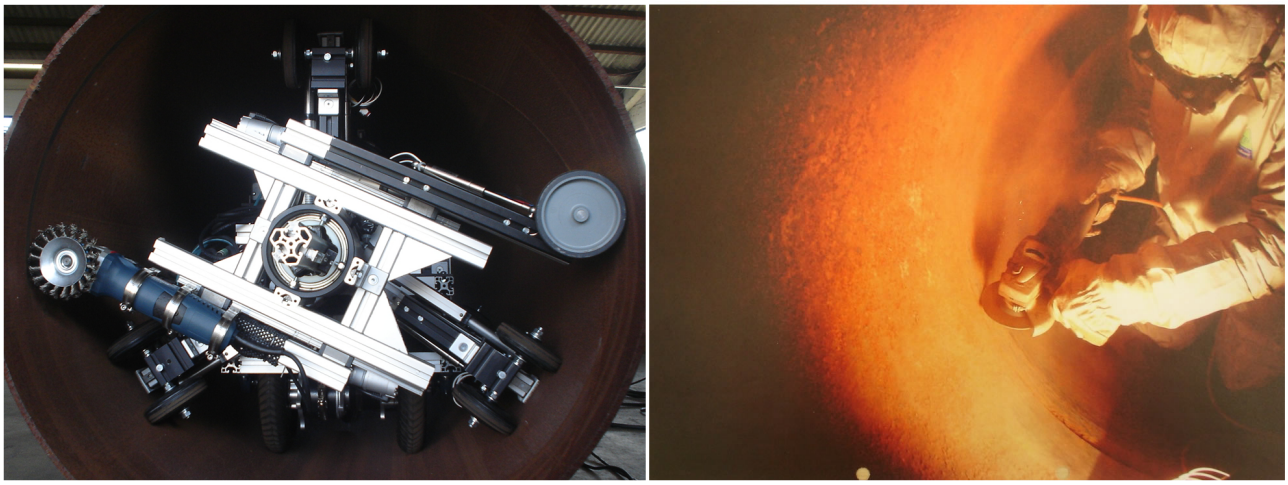


Fig. 1. DeWaLoP robot fixed inside the pipe, creating a rigid structure from its six wheeled-legs, cleaning with an angle grinder - wire brushes disk (left). Human operator inside the pipe, cleaning the pipe wall with an angle grinder - cutting disk (right).

2) Umbrella mechanism: The umbrella mechanism consists of a structure able to increase its height in order to adapt to different pipe diameters. The cleaning system is similar to an umbrella kind open-and-close mechanism, which makes the robot highly adaptable to different pipe sizes [10].

Commercial cleaning system such as Robocutter [11], KASRO robot [12], OptiCut [13] and IMS Turbo cutter [14] are smaller robots but use similar design like the umbrella mechanism. Lacking of stability, due to the push back effects and vibration caused in the cleaning process.

C. Tool-based cleaning - Cutting methods

1) Cutter cleaner arm: N. T. Thinh [15] describes an in-pipe robot for cleaning and inspecting, in which the cleaning mechanism is an arm consisting of small cutting plates. The arm is located on the front of the robot, perpendicular to the pipe's horizontal with the same length as the inner-pipe's diameter. From this configuration, the cleaning method consists of rotating the arm, milling all corrosion while the robot moves inside the pipe. Although the mechanism is able to remove strongly incrusted corrosion. The drawback of this configuration is the low flexibility of the cleaning tool to pipe displacement. In other words, the cleaning mechanism will damage the pipe if the pipes are not perfectly aligned.

C. D. Jung [16] proposes an in-pipe cleaning robot with the 6-link sliding mechanism which can be adjusted to fit into the inner face of the pipe using pneumatic pressure. The proposed in-pipe cleaning robot have self forward/backward movement as well as rotation movement of brush. However, the disk cleans all over the in-pipe wall without being able to focus on a specific area.

In contrast to the state-of-the-art cleaning mechanisms, DeWaLoP in-pipe robot is able to fix itself stably in a specific location using self suspension system. Independently from the main body of the robot, the cleaning tool is flexibly configured which can be adjusted in a cylindrical $3D$ space, able to move up to $100mm$ in the pipe's horizontal axis, and reach to the surface of the pipe with diameter in the range of $800mm$ to $1000mm$.

Besides its high positioning capability and flexibility, the cleaning mechanism is able to overcome vibrations and jump back forces from the cleaning tool with its integrated suspension system, mimicking the reaction of a human operator when such events happens.

III. DeWaLoP In-pipe Robot System

The DeWaLoP robot is intended to be a low cost robot with high reliability and easiness in use. The robot system includes a conventional in-pipe inspection system, which is carried out by using a cable-tethered robot with an onboard video system. An operator remotely controls the movement of the robot.

The robot consists of three main subsystems: control station, mobile robot, maintenance system, as shown in figure 2:

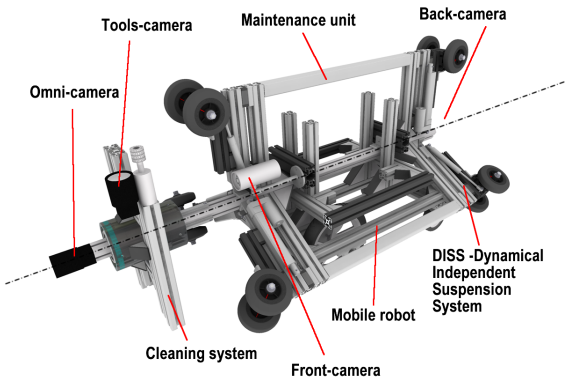


Fig. 2. DeWaLoP in-pipe robot perspective view.

A. Control station.

The control station monitors and controls all the components of the in-pipe robot. The controller includes a slate

computer for monitoring and displaying the video images from the robot's Ethernet cameras. Additionally, several 8 bits micro-controllers with Ethernet capabilities are included, sending and receiving commands to the in-pipe robot from the remote's joysticks and buttons [17].

B. Mobile robot.

The mobile platform is able to move along the pipe, carrying on board the electronic and mechanical components of the robot, such as motor drivers, power supplies, and etc. It uses a differential wheel drive which enable the robot to promptly adjust its position to remain in the middle of the pipe while moving.

C. Maintenance unit.

The maintenance unit consists of a wheeled-leg structure able to extend or compress with a Dynamical Independent Suspension System (DISS) [18]. When extending its wheeled-legs, it creates a rigid structure inside the pipe, so the robot tools work without any vibration or involuntary movement from its inertia. When compressing its wheeled-legs, the wheels become active and the maintenance unit is able to move along the pipe by the mobile robot.

The maintenance unit structure consists of six wheeled-legs, distributed in pairs of three, on each side, separated by an angle of 120°, supporting the structure along the centre of the pipe, as shown in figure 2. The maintenance unit combines a wheel-drive-system with a wall-press-system, enabling the robot to operate in pipe diameters varying from 800*mm* to 1000*mm*. Moreover, the maintenance unit together with the mobile robot form a monolithic multi-module robot, which can be easily mounted/dismounted without the need of screws [19].

The maintenance unit, includes two subsystems: Vision system and cleaning tool system.

Maintenance unit - Vision system. The in-pipe robot includes four cameras, in order to navigate in the pipe, detect defects and redevelop specific areas. For the navigation stage, two cameras are required, one located at the front, to inspect the way in the pipe, whereas the second located at the back, to inspect the way out. For the detection stage, an omni-directional camera is located at the front-end of the robot enabling the pipe-joint detection [20]. Finally, for the redevelopment stage, another camera is mounted on the cleaning mechanism. This camera acts as the human operator eyes, enabling the operator to follow the details of the redevelopment process.

IV. DeWaLoP Cleaning Mechanism

The concept of the DeWaLoP cleaning mechanism is based on the cylindrical robot principle, able to rotate along its main axes forming a cylindrical shape. The robot arm is attached to the slide so that it can be moved radially with respect to the column.

However, the DeWaLoP mechanism modifies the standard cylindrical robot into a double cylindrical robot, where both arms are connected to the central axis and opposite each other. In this configuration, an extra actuator is added to the standard model to extend/compress this second arm. Contrary to the standard cylindrical robot, the location of the rotating actuator is not in the central axis of the robot, it is located on the arm which is opposite to the arm with the tool, as shown in figure 3a. Hence, this rotating actuator (drive wheel) rotates and with it the entire cleaning mechanism.

If relating the forward transformation of the cylindrical robot to the DeWaLoP mechanism, the angular motion is modified. The forward transformation for a cylindrical robot is quiet simple, because it is equivalent to the transformation from a cylindrical to a Cartesian frame [21]. This frame moves in $3D$ space with the following dependencies on joint motions $\varphi(t)$, $r(t)$ and $z(t)$. Where $\varphi(t)$ is the joint revolute, $r(t)$ is the length of the arm, $z(t)$ is the axial distance in the z-axis (pipe's horizontal) and A represents the width of the linear and circular bearings of the mechanism on the central axis of the robot.

$$x_0(t) = [A + r(t)]cos(\varphi(t)) \tag{1}$$

$$y_0(t) = [A + r(t)]sin(\varphi(t)) \tag{2}$$

$$z_0(t) = z(t) \tag{3}$$

Likewise, the rotation ratio of the DeWaLoP mechanism is given by the pipe radius r_{pipe} divided to the radius of the drive wheel r_{wheel}, similar to a planetary gear, where the outer gear (the drive wheel) revolve from the central (the pipe). Thus, the revolutions needed from the drive wheel (dw_r) to complete a full rotation is $dw_{r360^\circ} = r_{pipe}/r_{wheel}$.

In this way, we can use dw_r (drive wheel revolutions) to represent the cylindrical robot angular value.

$$\varphi(t) = (dw_r/(r_{pipe}/r_{wheel})) \times 360^\circ \tag{4}$$

And the forward transformation for the model is:

$$x'_0(t) = x_0(t) = x(t) \tag{5}$$

$$y_0(t) = [(A/2) + r(t)]cos(\varphi(t)) \tag{6}$$

$$y'_0(t) = [(A/2) + r'(t)]cos(\varphi(t) + 180^\circ) \tag{7}$$

$$z_0(t) = [(A/2) + r(t)]sin(\varphi(t)) \tag{8}$$

$$z'_0(t) = [(A/2) + r'(t)]sin(\varphi(t) + 180^\circ) \tag{9}$$

where (x_0, y_0, z_0) are the coordinates for the cleaning tool arm and (x'_0, y'_0, z'_0) are the coordinates for the drive wheel arm, $\varphi(t)$ is the joint revolute, $r(t)$ is the length of the arm with the cleaning tool, $r'(t)$ is the length of the arm with the drive wheel, $x(t)$ is the axial distance in the x-axis (pipe's horizontal) and A represents the width of the linear and circular bearings of the mechanism installed on the central axis of the robot.

In this simplified robot configuration, the cleaning tool is located opposite to the drive wheel and perpendicular to the robots central axis, as shown in figure 3a. Therefore, the spacing to attach power tools is considerable small. The maximum cleaning tool height is given by:

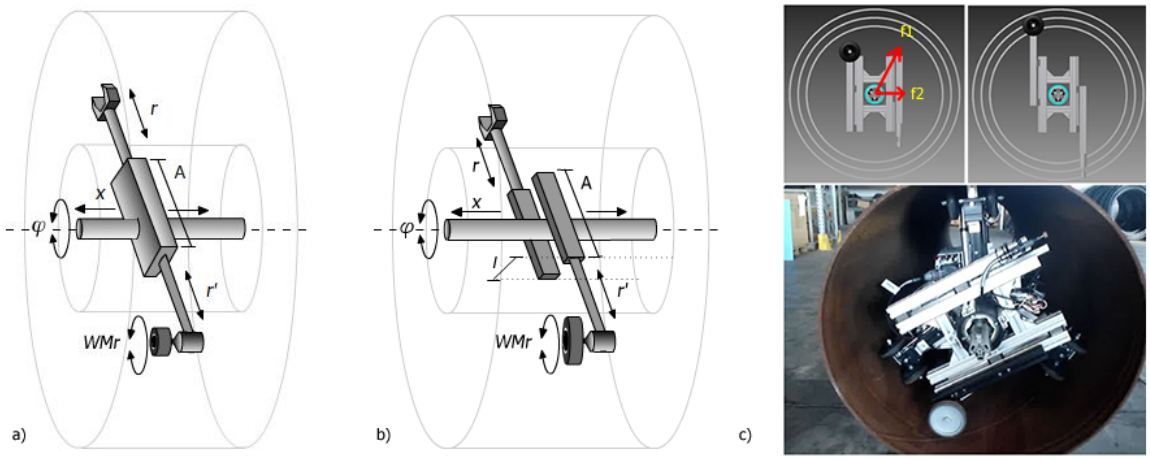


Fig. 3. a) DeWaLoP simplified model (double cylindrical robot). The segments r and r' can be compress or extend by a linear actuator, the segment A is fixed, as it integrates a combination of linear and circular bearings enabling the mechanism to cover $3D$ space. b) Simplified H -configuration model. Similar to the double cylindrical robot model, with the difference that the robot arms are translated to a distance l from the robot's central axis. c) Linear actuators working axis in H -configuration. d) Maximum extension of the arms reaching $1000mm$ diameter pipe (top); maximum compression of the arms with length of $500mm$ (bottom).

$$CTh_{max} = Pr - \frac{A}{2} - Dtpr \tag{10}$$

where Pr is the pipe radius and $Dtpr$ is the clearance distance of the mechanism to the pipe wall. In other words, the mechanism in this configuration restricts the height size of the attached cleaning tool to $CTh_{max} \leq 300mm$. That means, only angle grinders with $disks \leq 115mm$ may be attached [22].

H- Configuration

In order to attach bigger power tools to the cleaning mechanism, with heights in the range up to $500mm$, a new configuration is presented.

In this configuration the arms are not mounted directly over the bearing arrangement in the main axis. Instead, they are translated to its sides, around it and parallel to each other, as shown in figure $3b$. In this way, the maximum height of the cleaning tool is determined by the "H" geometric configuration, as shown in figure $3c$.

$$CTh_{maxH} = sin(\alpha)D - Dtpr \tag{11}$$

where $\alpha = tan^{-1}(H/W)$ is the angle between the direction vector $f1$ which is from the pipe center to the cleaning tool mounted at the end of the arm, to the direction $f2$ which is perpendicular to the arm, H is the height of the tool system in compress mode ($H = 500mm$), W is the width of the tool system ($W = 300mm$) and D is the diameter of the pipe ($D = 800mm$).

Consequently, it is possible to attach cleaning power tools with height up to $CTh_{maxH} \leq 685.99mm - Dtpr$, such as straight grinders and angle grinders with $125mm$ disks or bigger.

From the simplified H -configuration model, where l is the shifted distance of the arms from the central axis of the robot, as shown in figure $3b$. The position of the end effector from the robot cleaning tool is $p = (p_x p_y p_z)^T$ and the position of the drive wheel is $p' = (p'_x p'_y p'_z)^T$, where $p_x = p'_x$ represents the translation in the pipe's horizontal axis.

The robot's cleaning tool direct kinematics is

$$p = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = \begin{pmatrix} 1 \\ l/2\sin\varphi + r\cos\varphi \\ l/2\cos\varphi - r\sin\varphi \end{pmatrix} \tag{12}$$

and the robot's drive wheel direct kinematics is

$$p' = \begin{pmatrix} p'_x \\ p'_y \\ p'_z \end{pmatrix} = \begin{pmatrix} 1 \\ l/2\sin(\varphi + 180^\circ) + r'\cos(\varphi + 180^\circ) \\ l/2\cos(\varphi + 180^\circ) - r'\sin(\varphi + 180^\circ) \end{pmatrix} \tag{13}$$

Stability analysis of the H -configuration mechanism

The vibrations and jump back forces from the cleaning tool while removing corrosion may appear in various directions. The forces in the pipe's horizontal (x - axis) are damped by the cleaning tool structure. These are the most noxious forces affecting the robot, because no suspension system are damping it. However, these forces are unlikely to occur when using angle grinders, as shown in figure 1. Due to the rotation of the cleaning tool, which is similar to the mechanism of the drive wheel, affecting only the y and z axis, reducing x - axis forces. Nevertheless, these types of noxious forces appear when using straight grinders, in this case, the cleaning head is rotating over the pipe's surface resulting in vibration forces over all direction.

The vibrations and forces perpendicular to the cleaning tool, in the y - axis, are damped by the drive wheel located on the opposite end of the mechanism. In this case, the damping degree is given by the friction coefficient μ of the drive wheel to the pipe's surface and the force applied by the wheel to the surface. This may vary if the surface is wet or dry, with friction coefficient in the ranges from $\mu = 0.35$ to $\mu = 0.5$. And the damping forces are in the range of $140N$ to

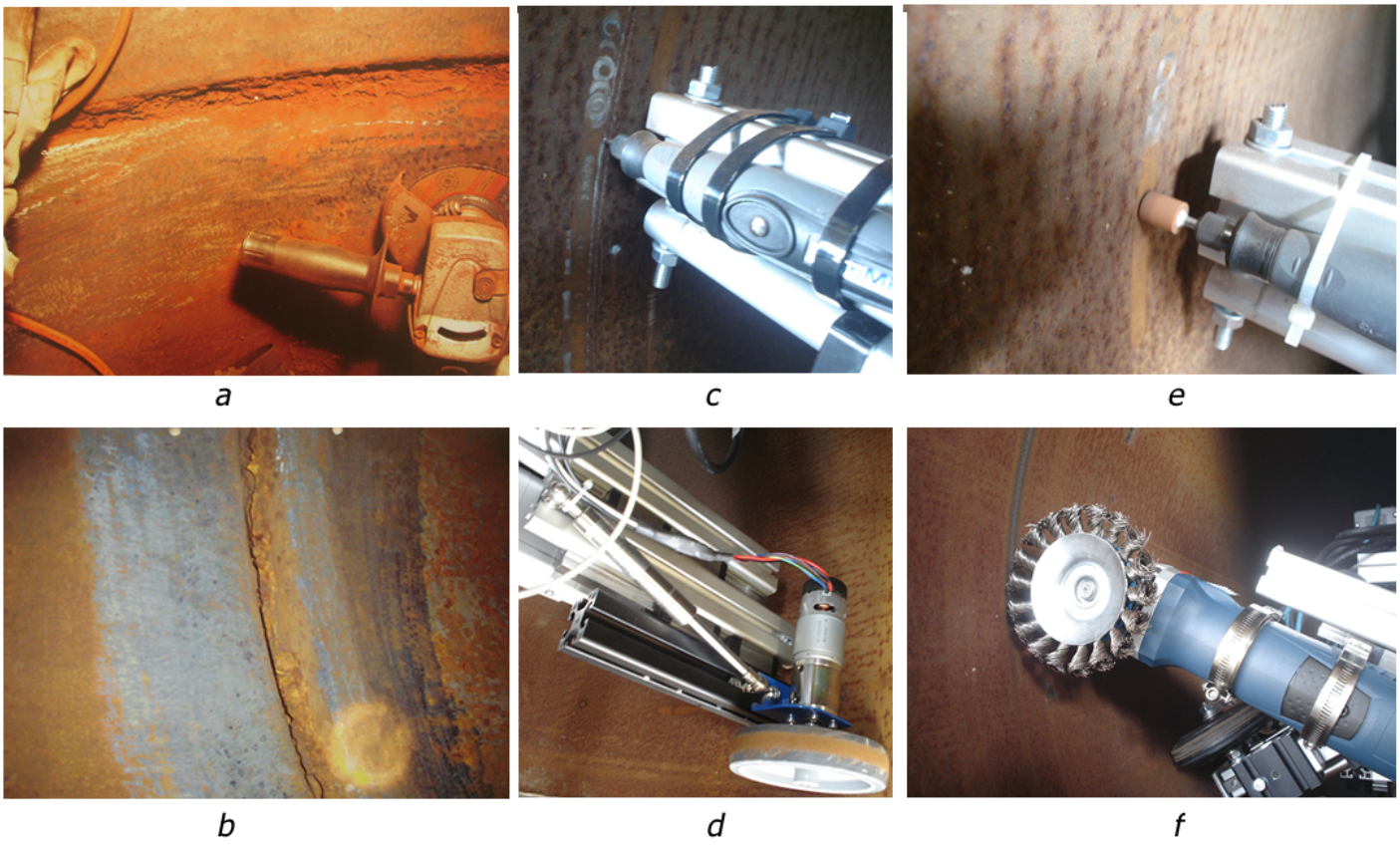


Fig. 4. a) Human operator partial cleaning. b) Cleaning result from human operator c) DeWaLoP robot with milling head. d) Drive wheel with suspension system. e) DeWaLoP robot with straight grinder - grinding head installed. f) DeWaLoP robot with angle grinder - brushes disk installed.

200N respectively, with a 400N spring on the drive wheel. In other words, if vibrations or jump back forces in the y - axis stronger than 200N are affecting the cleaning tool, then the drive wheel may slip a bit, mimicking the arms of a human operator damping these forces.

The vibrations and jump back forces opposite to the cleaning tool, in the z - axis, are damped by its 400N suspension system. Mimicking again, a human operator retracting the arms when opposite forces from the application axis appear.

V. Experiment Analysis

We use a cast-iron pipe segment provided by Vienna Waterworks Company to perform the experiment. The pipe segment has length of 6000mm and diameter of 900mm.

A. Procedure of the experiment

The process of conducting the experiment is described as following (some photos taken from this process are shown in Fig 4):

Step 1: The mobile robot and the maintenance unit were arranged as a monolithic robot. The selected cleaning tool was mounted on the cleaning mechanism.

Step 2: The robot was moved inside the pipe and stopped over a determined point. The robot adapted its maintenance unit configuration, from compressed to extended mode, so that the wheeled-legs formed the robot to be a centered and rigid structure inside the pipe.

Step 3: Once the extending is finished, the cleaning mechanism is enabled and selected. The arm with the drive wheel is extended first until it makes proper contact to the pipe surface, by compressing its spring (400N). Then the arm with the cleaning tool is moved to the desired position in cylindrical 3D space.

Detailed information of the DeWaLoP cleaning mechanism and cleaning process, such as simulation and videos, can be found in [23].

B. Experimental results and evaluation

The job of cleaning the inner surface of a pipe with diameter of 1000mm and length of 30mm, which is an area of size $a = \pi \times D \times w = 0.094m^2$, will cost a human operator between 30 to 60 minutes, according to the estimation of Vienna waterworks. In our experiment, the DeWaLop robot completed the job within 5 to 15 minutes, and the corrosion is fully removed. The actually speed highly depends on the displacement of the pipe. For a pipe without displacement, the cleaning tool revolves without adjusting its position according to the pipe horizontal direction, so that it is able to achieve the fastest cleaning speed of 5-7 minutes. While for pipes with displacements, the cleaning mechanism is

required to adjust its position and the cleaning process takes up to 15mins.

Hence, the cleaning speed of DeWaLop robot is 6 to 10 times faster than human operators. Besides, the cleaning pattern of DeWaLoP is mostly a straight line, in contrast to the zig-zag pattern left by human operators.

VI. Conclusion

A new in-pipe cleaning mechanism was presented and compared to the state-of-the-art cleaning systems. The presented mechanism improves the state-of-the-art in multiple aspects, such as enabling the cleaning tool to precisely step on the pipe surface by using the double cylindrical H - configuration mechanism in the robot arm, and protecting the pipe from noxious forces caused by the cleaning tool from its integrated suspension and drive wheel configuration.

The presented mechanism, modifies the cylindrical robot configuration in to a double cylindrical robot. In which the angular actuator is located on one of the arms and not in the central axis as the classical model. Enabling the mechanism to rotate with high precision similar to a planetary gear. While rotating the system removes the corrosion of the pipe with a cleaning tool mounted on the opposite arm to the drive wheel.

Cleaning results from the DeWaLoP mechanism are presented and compared to human performances, revealing a faster and accurate trajectory of the cleaning path.

ACKNOWLEDGMENT

This work is part-financed by Project DeWaLoP from the European Regional Development Fund, Cross- Border Cooperation Programme Slovakia- Austria 2007-2013.

References

[1] e. a. S. Burn, "Pipe leakage future challenges and solutions," in *Pipes Wagga Wagga Conference*, 1999.

[2] e. a. O. Hunaidi, "Detecting leaks in plastic pipes," *Journal of the American Water Works Association 21st Century Treatment and Distribution*, vol. 92, no. 2, pp. 82–94, 2000.

[3] S. L. V. Archodoulaki, G. Kuschnig and M. Werderitsch, "Silane modified polyether sealant failure in drinking water pipes," in *MoDeSt*, 2010.

[4] S. J. S. Yang and S. Kwon, "Remote control system of industrial field robot," in *IEEE International Conference on Industrial Informatics*, 2008.

[5] A. Amir and Y. Kawamura, "Concept and design of a fully autonomous sewer pipe inspection mobile robot kantaro," in *IEEE International Conference on Robotics and Automation*, 2007.

[6] S. Roh and H. Ryeol, "Differential-drive in-pipe robot for moving inside urban gas pipelines," in *IEEE transactions on robotics*, 2005.

[7] C. Z. B. Rajani and S. Kuraoka, "Pipe-soil interaction analysis of jointed water mains," *Canadian Geotechnical Journal*, vol. 33, no. 3, pp. 393–404, 1996.

[8] J. Saenz, N. Elkmann, T. Stuerze, S. Kutzner, and H. Althoff, "Robotic systems for cleaning and inspection of large concrete pipes," in *Applied Robotics for the Power Industry (CARPI), 2010 1st International Conference on*, oct. 2010, pp. 1 –7.

[9] V. G. H. Schempf, E. Mutschler and W. Crowley, "Grislee: Gasmain repair and inspection system for live entry environments," *The International Journal of Robotics Research*, vol. 22, no. 7-8, pp. 603–616, 2003.

[10] J. Z. Z. X. Li, "Development of the self-adaptive pipeline cleaning robot," in *Advanced Materials Research*, 2010, pp. 97–101.

[11] KATEPMO, *http://www.kate-pmo.ch/index.php*. KATEPMO, 2012. [Online]. Available: http://www.kate-pmo.ch/index.php

[12] Prokasro, *http://prokasro.de/*. Prokasro, 2012. [Online]. Available: http://prokasro.de/

[13] Optimess, *http://www.optimess.com/*. Optimess, 2012. [Online]. Available: http://www.optimess.com/

[14] IMSRobotics, *http://www.ims-robotics.de/en/produkte.html*. IMSRobotics, 2012. [Online]. Available: http://www.ims-robotics.de/en/produkte.html

[15] N. T. Thinh, N. Ngoc-Phuong, and T. Phuoc-Tho, "A study of pipe-cleaning and inspection robot," in *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, dec. 2011, pp. 2593 –2598.

[16] C. D. Jung, W. J. Chung, J. S. Ahn, M. S. Kim, G. S. Shin, and S. J. Kwon, "Optimal mechanism design of in-pipe cleaning robot," in *Mechatronics and Automation (ICMA), 2011 International Conference on*, aug. 2011, pp. 1327 –1332.

[17] L. Mateos, M. Sousa, and M. Vincze, "Dewalop remote control for in-pipe robot," in , *2011 15th International Conference on Advanced Robotics (ICAR)*, june 2011, pp. 518 –523.

[18] L. A. Mateos and M. Vincze, "Dewalop robot dynamical independent suspension system," in *ICMET*, 2011, pp. 287–292.

[19] ——, "Dewalop-monolithic multi-module in-pipe robot system," in *ICIRA*, 2011, pp. 406–415.

[20] ——, "Dewalop - robust pipe joint detection," in *IPCV*, 2011, pp. 727–732.

[21] S. Kucuk and Z. Bingul, "The inverse kinematics solutions of industrial robot manipulators," in *Mechatronics, 2004. ICM '04. Proceedings of the IEEE International Conference on*, june 2004, pp. 274 – 279.

[22] L. A. Mateos, A. Rakos, and M. Vincze, "Dewalop in-pipe redevelopment system design," in *ARW*, 2012, pp. 101–106.

[23] D. C. M. Animations and Videos, *DeWaLoP Cleaning Mechanism Animations and Videos*. AM Computer Systems, 2013. [Online]. Available: http://www.amcomputersystems.com/AM/research/robotics/d.html

Improving the ROS Arm Navigation Stack by Using Stochastic Inverse Kinematics

Clemens Mühlbacher, Gerald Steinbauer, Michael Reip and Stephan Gspandl

***Abstract*— Mobile manipulation becomes more and more of a must to cope with industry's need for more flexible and adaptive solutions in manufacturing and production. This paper deals with an important subtask of mobile manipulation. It presents a novel approach to solve the inverse kinematic problem. This approach contributes to more universal applicability of mobile manipulation algorithms. In further consequence, it thus helps to promote the application of mobile manipulation in industrial settings.**

In this paper, we treat the inverse kinematic problem as an optimization problem and solve it with the help of a global optimization algorithm. The performance of the algorithm concerning success as well as accuracy is evaluated in several experiments. It is shown that in contrast to other algorithms our approach is always able to retrieve a solution if existing. This comes, though, with higher costs concerning performance.

I. Introduction

Mobile manipulation is a hot topic in the industry as well as robotic research community. Robot arms executing paths statically to solve identical problems in a repetitive way are widely known. One can find many examples, especially in the automobile industry. There, manipulators have been used to move, assemble, weld or coat cars or car parts. In the last few years, though, one could notice the beginning of a shift. Robot arms were not exclusively deployed for simple tasks in a static manner, but instead are employed in mobile robots to enhance their field of operation. Such robot systems are far more flexible than their static friends. As [1] puts it, the market demands for new agility and flexibility (mainly due to globalization, explosion of variety, a shift to customized production and the necessity to change the types of products fast and scale production), but fixed robots are unable to live up to these expectations. The authors argue that industrial mobile manipulation is the technology which meets the need of transformable production systems.

In this area of robotics the Robot Operating System (ROS)[2] is a widely used framework to perform such complex tasks. Examples for mobile manipulation tasks can be found in [3] (see also Figures 1). To perform simple as well as complex manipulation tasks under different robot settings, ROS provides a wizard to configure the necessary object manipulation tools [4]. These tools form a chain of solvers for the various problems, which occur during a manipulation task. The first step within the chain is the inverse kinematic (IK), which is a mapping from the Cartesian space to the control parameters of the robot arm. To solve the IK a general method is used to cope with the different robot arms. This is a major difference to industrial robots, where a general way to solve the inverse kinematic is not used. Instead, inverse kinematic solvers are employed that are specialized to work with one particular manipulator. Such an approach is infeasible in research, because one had to be able to deal with a variety of new hardware in science. One major drawback of the general solving method, which is used per default within ROS for the IK, is that it might not be able to find a solution even if it exists. This can cause the complete manipulation task to fail. Thus, general algorithms are preferred which will find a solution if existing. Such algorithms allow for a much more reliable and robust execution of manipulation tasks.

In this paper, we present a novel general inverse kinematic solver which is able to find a solution for a much larger set of problems compared to other techniques.

In the following, we will discuss related work which deals with general inverse kinematic solving methods. In the second chapter, we will define the inverse kinematic problem formally. After this chapter we explain our algorithm to solve the inverse kinematic problem. In Chapter V we will present the results of a series of experiments which were performed to demonstrate the general applicability as well as the performance of the algorithm. In the last chapter, we draw some conclusions about the presented approach.

Fig. 1. Manipulation task. Image is taken from [3]

II. Related Work

There are different methods to solve the inverse kinematic problem in a general way. Before we present our novel approach to solve the inverse kinematic problem we will briefly

C. Mühlbacher and G. Steinbauer are with the Institute for Software Technology, Graz University of Technology, Graz, Austria. M. Reip and S. Gspandl are with incubedIT, Graz, Austria. {cmuehlb,steinbauer}@ist.tugraz.at, {s.gspandl, m.reip}@incubedit.com

discuss some approaches and point out their weaknesses.
The first method we will discuss is an analytic solver. This solver is generated with the help of the Ikfast [5] algorithm, which is a part of the openRave system [5]. The approach creates a system of equations from the forward kinematic equations of the robot arm. In order to solve the system variables are ranked and chosen according to their rank. The fewer solutions a variable allows, the higher it is ranked. If there are multiple possible choices to solve the system with any of the chosen variables, ikFast chooses the one with the fastest numerical operations. This results in a fast analytic solver, which prefers simple solutions with fast numerical operations. A major drawback of this approach is that the generation of the solver can take a long time or, in some cases, a solver cannot be generated at all. Also if a solver is generated the solver does not always find a solution, which is a result of the implementation of this approach and not of its theoretical background. In order to use this generated solver in ROS a simple wrapper is used. The wrapper changes the request in such a way that a bridge between ROS and openRave can be used. If the algorithm cannot find a solution the wrapper slightly changes the position and orientation, to find a solution near the desired one.
Another method to find a solution for the inverse kinematic problem is a Newton-Raphson inverse kinematic solver (some examples can be found in [6], [7]). This kind of solver uses Jacobian matrices. Jacobian matrices describe the relation between joint and link velocities, which have to be positioned within Cartesian space. The first step of the approach calculates of the velocity within the Cartesian space of the link, which has to be positioned. The calculation is based on the difference between current and desired link's pose. Using the inverse of the Jacobian matrix the velocity within the Cartesian space is mapped onto a joint velocity for each joint of the robot arm. These joint velocities are used to update the current joint values. If the current joint values move the link near to the desired pose, the procedure returns with the current joint values as a result, otherwise the procedure starts again calculating the Cartesian velocity. This method suffers from two drawbacks, which can prevent the algorithm from finding a solution even if one exists. The first drawback is that the inverse of the Jacobian matrix is ill conditioned near a singularity. The second drawback are the initial joint values. The joint values can be chosen in such a way that the algorithm does not converge to a solution. To overcome this problem the algorithm is called multiple times with different initial joint values to find a solution. The implementation we use is KDL [1], which offers the possibility to interact with ROS and is the standard inverse kinematic solver [4].

III. Problem Formulation

Before we describe our approach for solving the inverse kinematic problem we will define the inverse kinematic problem formal. We assume a robot arm (RA) as a set of links $L = \{l_1, \dots, l_m\}$ which are connected through a set of joints $J = \{j_1, \dots j_{m-1}\}$. The robot arm moves in the Euclidean space. The motion space consists of a translation $\mathbb{R}^3$ and a rotation $SO(3)$ part. The motion in space is the combination of this two parts $SE(3) = \mathbb{R}^3 \times SO(3)$. Each link $l_i \in L$ has its own coordinate frame. The pose $\mathbb{P}^i$ of this coordinate frame can be calculated with the forward kinematic function $\mathbb{P}^i = FK^i(v_{j_1}, \dots, v_{j_{i-1}})$, where $i = 1 \dots m$ and v_{j_i} is the joint value of j_i. The inverse kinematic problem form a mapping $SE(3) \to \mathbb{R}^{m-1}$ and is defined as follows: given a robot arm (RA), a desired position (P) and orientation (O) of a link $l_i \in L$, find for each joint $j_i \in J$ a value such that that $position(\mathbb{P}^i) = P$ and $orientation(\mathbb{P}^i) = O$. Within a robotic system additional constraints have to be considered. The first constraints are the joint limits $\forall_{j_i \in J} : j_i^{min} < j_i < j_i^{max}$. Additionally, no link of the robot arm should collide with any other link of the robot arm or any object o of the known environment KE. To check this constraint we need the volume of a link. The volume of the link L_i in $\mathbb{R}^3$ is given through $L_i(v_{j_1}, \dots, v_{j_{i-1}})$. With this definition the collision avoidance can specified as $\forall_{i=1\dots m} : \big(\forall_{o \in KE} : L_i(v_{j_1}, \dots, v_{j_{i-1}}) \cap o = \varnothing\big) \wedge \big(\forall_{k:1\dots m} : k \neq i \wedge L_i(v_{j_1}, \dots, v_{j_{i-1}}) \cap L_k(v_{j_1}, \dots, v_{j_{k-1}}) = \varnothing\big)$.

IV. Stochastic Inverse Kinematics

In order to solve the inverse kinematic problem with all the additional constraints we propose an inverse kinematic solver, which uses a global optimization algorithm. This is a very general idea and a wide variety of different optimization algorithms can be applied. To define the optimization problem we will introduce some functions that are used later. To calculate a distance we use two different functions. To calculate the position difference *posDiff* we use the squared Euclidean distance in $\mathbb{R}^3$ shown in Equation 1 .

$$posDiff(p_1, p_2) = \|p_1 - p_2\|^2 \tag{1}$$

The second function *orDiff* calculates the orientation error of two orientations, which are specified via quaternion's.

$$orDiff(q_1, q_2) = |1 - (q_1 \cdot q_2)^2| \tag{2}$$

The objective function of the optimization problem for a link i is defined as:

$$\begin{aligned} F^i(v_{j_1}, \dots, v_{j_{i-1}}) =& \alpha * posDiff(FK^i(v_{j_1}, \dots, v_{j_{i-1}}),\ P) \\ &+ \beta * orDiff(FK^i(v_{j_1}, \dots, v_{j_{i-1}}),\ O)] \end{aligned} \tag{3}$$

α and β are weights which are used to scale the different parts according to the used measurements. For example the result of the *orDiff* function is between [0,1], whereas *posDiff* use meters. Thus the orientation difference is scaled, which is necessary to force the algorithm to get a correct position and not only a correct orientation. The optimization problem, to position link l_i, is stated as follows:

$$\min_{v_{j_1}, \dots, v_{j_{i-1}}} F^i(v_{j_1}, \dots, v_{j_{i-1}}) \tag{4}$$

[1] http://www.orocos.org/kdl

With this optimization problem the inverse kinematic can be solved with any optimization algorithm. For example a similar approach was also used in [8]. We use a global optimization algorithm to avoid problems that can occur with local minima. A further extension to [8] are continuous values. It is possible to use different global optimization algorithms, which fulfill the requirements to solve this problem. We will use the *Differential Evolution Algorithm* [9]. To understand how the algorithm works we will define two terms and their representation within the algorithm. The first term is the *individual* representing a possible solution of the problem. To represent such an *individual* within the algorithm it consists of two parts, the joint values of the robot arm and the value of the objective function. The second term is the *population*. It represents a list of possible solutions and is, within the algorithm, a list of *individuals*. Algorithm 1, uses different populations to find the minimum of the function. In order to find the minimum, the algorithm

Algorithm 1: *findMinimum*

input : F ...Objective function,
C ...set of constraint,
$maxIt$ ...maximum iterations of the algorithm,
CR ...cross over ration,
N ...size of the *population*
output: a set of possible solutions CP

$CP = generateValidPositions(N, C)$
$evaluate(CP, F)$
$i = 0$
while $(i < maxIt) \wedge \neg convergenceCriterialsMet(CP)$ **do**
  $donors = mutate(CP)$
  $trials = recombine(CP, donors, CR)$
  $trials = clamp(trials, C)$
  $evaluate(trials, OF)$
  $CP = select(CP, trials)$
  $i = i + 1$
end
return CP

first generates an initial *population*. These initial population represents random samples within the joint space. These samples are chosen in such a way that they fulfill the constraints. Afterwards for each *individual* the objective function is calculated and stored in the *individual*. Within the loop the first step is to mutate the current *population* (CP) to generate a new *population*, called $donors$. This mutation is produced through the combination of three *individuals* within the *population*. The mutation algorithm 2 uses the best and two random *individuals* with specified weights *f1* and *f2* to combine them into a new *individual*. After the $donors$ are created they are combined with the current *population* CP to generate the $trials$. The algorithm to combine the *populations* can be seen in Algorithm 3. The algorithm chooses randomly a joint value which is taken from the donor *individual*. Thus at least one joint value is taken from the

Algorithm 2: *mutate*

input : P ...Current *population*,
N ...size of the *population*,
f1 ...first combinatoric weight,
f2 ...second combinatoric weight
output: a new *population* $donors$

for $i = 1 : N$ **do**
  $r_1 = drawRandomIndividual(CP)$
  $r_2 = drawRandomIndividual(CP)$
  $best = getBestIndividual(CP)$
  for $j = 1 : k$ **do**
    $newIndividual[j] = f1 * (r_1 - r_2) + f2 * (best - r_1)$
  end
  $donors = donors \cup newIndividual$
end
return $donors$

donor *individual*. The other joint values are chosen randomly from the donor *individual* or the current *individual*, according to the CR constant. After the $trials$ are calculated each

Algorithm 3: *recombine*

input : P ...Current *population*,
$donors$ ...donor *population*,
CR ...Cross over ration defining how individuals are combined,
N ...size of the *population*
output: a new *population* $trials$

for $i = 1 : N$ **do**
  $index = drawUniformFrom([1 \dots k])$
  for $j = 1 : k$ **do**
    if $(j = index) \vee (drawUniformFrom([0 \dots 1]) \leq CR)$ **then**
      $individualToCombine = donors[i]$
    else
      $individualToCombine = P[i]$
    end
    $newIndividual[j] = individualToCombine[j]$
  end
  $trials = trials \cup newIndividual$
end
return $trials$

individual of the trials is checked if the *individual* fulfills the constraints. If a constraint is violated the joint values are clamped in such a way that the constraints are satisfied. The resulting $trials$ are valid possible solutions and each *individual* out of the $trials$ is evaluated. After the evaluation step the next step is to select the best *individuals* from P and the $trials$. This is achieved through Algorithm 4. To generate P', the algorithm checks an *individual* from P and the $trails$, which are at the same position within their

Algorithm 4: *select*

input : P ...Current *population*,
$trials$ ...trial *population*,
N ...size of the *population*
output: a new *population* P'

```
for i = 1 : N do
    if (objective(P[i]) < objective(trials[i])) then
        newIndividual = P[i]
    else
        newIndividual = trials[i]
    end
    P' = P' ∪ newIndividual
end
return P'
```

population. The *individual* with a smaller objective is part of the new *population*. After P' is generated it is saved as the current *population* and the algorithm checks if the *population* converges. The convergence check is simple a test if a specified amount of individuals is close enough to the desired pose. In the current implementation the population converges if 10 % of the *population* is close enough to the desired pose.
In order to find a solution for an inverse kinematic problem the Algorithm 1 is called and a solution is chosen in such a way that no collision occurs. If none of these solutions is possible, the algorithm is called again. A timeout is used to bound the number of possible recalls. After some recalls the number of *individuals* or the number of function calls per *individual* are increased. Recalling this procedure is used to force the algorithm to find a solution even if there are many solutions that would result in a collision, e.g. within a cluttered environment.

V. Experimental Results

In order to demonstrate the capabilities of the algorithm we perform two sets of evaluations.
The first evaluation [2] is used to test the inverse kinematic in a general way with many different target poses. To perform this evaluation first a random joint configuration is sampled. The configuration is checked if it does not result in a self-collision of the robot arm. If this check is passed the resulting pose of the last link of the robot arm is calculated with the help of a forward kinematic algorithm. After calculating the resulting pose the inverse kinematic is called to find a solution. To measure the performance the time, position and orientation error as well as the success to find a solution are recorded. The tests are performed with two different popular research robot arms, namely the Katana 400 6m180 (see Figure 4) and the KUKA youBot (see Figure 6). In the test we use three different inverse kinematic solvers: KDL, openRave and the newly presented solver (which is called stochastic solver). The parameter of the stochastic solver was set as follows: Initial N was set to 100 $maxIt$ was set to 200. After $findMinimum$ was recalled 5 the N was increased by a factor of 1.2 and $maxIt$ was increased by a factor of 1.5. The CR is set to 0.8. This parameters were empirically evaluated. In order to minimize the initialization's influence on the performance each pose is tested 10 times. A pose is counted as a success, if the position difference is lower than 0.01 and the orientation difference is smaller than 0.1. Equation 5 is used to calculate the success rates.

$$success\ rate = \frac{\#Found\ solutions}{\#test} \tag{5}$$

To calculate the block success rate a block is counted as successful if there is at least one solution found within this block. The bock success rate itself is calculated through Equation 6. The success rate as well as the block success rate can be seen in table I.

$$block\ success\ rate = \frac{\#Successful\ blocks}{\#blocks} \tag{6}$$

To calculate the position error Equation 1 is used. The resulting position errors, which can be seen in Table II, are given in square meters. The recorded orientation error is calculated through Equation 2 and can be seen in Table III. The run time is recorded in seconds and can be seen in table IV and Figure 3.

Robot arm	IK solver	Success rate	Blocks success rate
Katana 400 6m180	KDL	0.8722	0.9844
Katana 400 6m180	OpenRave	0.9378	0.9376
Katana 400 6m180	Stochastic	1	1
Youbot	KDL	0.7911	0.9933
Youbot	OpenRave	- (*)	- (*)
Youbot	Stochastic	1	1

TABLE I
SUCCESS RATE OF THE DIFFERENT INVERSE KINEMATIC ALGORITHMS.
(*) IKFAST IS NOT ABLE TO GENERATE A SOLVER.

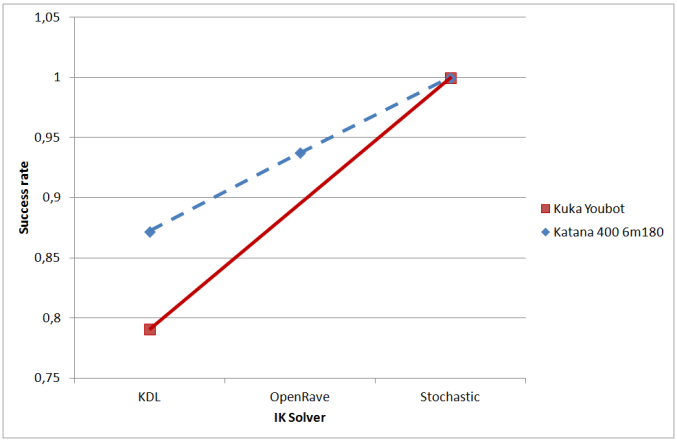


Fig. 2. Success rate of the different inverse kinematic algorithms

[2]The evaluation was performed on a Intel Core i7 Q 820, 1.73 GHZ with 8 GB of RAM. The operating system was a Ubuntu 12.04 32-bit. ROS Fuerte was used to perform the evaluation.

Robot arm	IK solver	Mean	Median	stdv
Katana 400 6m180	KDL	1.12e-11	2.40e-013	2.71e-11
Katana 400 6m180	OpenRave	5.63e-21	1.66e-24	1.15e-19
Katana 400 6m180	Stochastic	4.45e-5	1.30e-15	4.81e-4
Youbot	KDL	1.09e-11	9.77e-14	2.77e-11
Youbot	OpenRave	- (*)	- (*)	- (*)
Youbot	Stochastic	2.25e-4	5.89e-16	9.75e-4

TABLE II

POSITION ERROR OF THE DIFFERENT INVERSE KINEMATIC ALGORITHMS. (*) IKFAST IS NOT ABLE TO GENERATE A SOLVER.

Robot arm	IK solver	Mean	Median	stdv
Katana 400 6m180	KDL	2.39e-13	2.17e-19	2.11e-12
Katana 400 6m180	OpenRave	0.0203	2.17e-19	0.1185
Katana 400 6m180	Stochastic	9.12e-4	8.03e-14	0.0069
Youbot	KDL	3.09e-013	2.17e-19	2.20e-12
Youbot	OpenRave	- (*)	- (*)	- (*)
Youbot	Stochastic	0.0041	3.90e-14	0.0123

TABLE III

ORIENTATION ERROR OF THE DIFFERENT INVERSE KINEMATIC ALGORITHMS. (*) IKFAST IS NOT ABLE TO GENERATE A SOLVER.

Robot arm	IK solver	Mean	Median	stdv
Katana 400 6m180	KDL	0.0016	2.24e-4	0.0037
Katana 400 6m180	OpenRave	0.0344	0.0030	0.2167
Katana 400 6m180	Stochastic	0.1656	0.1580	0.0864
Youbot	KDL	0.0082	1.00e-003	0.0092
Youbot	OpenRave	- (*)	- (*)	- (*)
Youbot	Stochastic	0.3092	0.2930	0.0945

TABLE IV

RUNTIME OF THE DIFFERENT INVERSE KINEMATIC ALGORITHM. (*) IKFAST IS NOT ABLE TO GENERATE A SOLVER.

For the evaluation the input is a random pose and the expected output is the corresponding inverse kinematic. This is done 450 times for every algorithm. Notably, the stochastic solver always finds a valid solution if there is any (see Table I). In Figure 2 it can be seen that the stochastic solver achieves a higher success rate than any other solver. It is important to notice that openRave is not able to produce a solver for the KUKA youBot. The resulting position error (see Table II) and the orientation error (see Table III) is a bit higher. Though, the runtime of the very fast KDL cannot be achieved (see Table IV), the proposed algorithm is still preferable, due to its excellent success rate. This drawback can also be seen within the Figure 3, which demonstrate the run time of the different solvers. Another observation of the run time is that each solver finds faster a solution for the Katana 400 6m180 compared to the KUKA youBot.

The second evaluation was performed on real hardware. In order to test the accuracy as well as the success of the algorithm, we created a pick and place test environment (see Figure 5). The task was to pick up small balls, with a

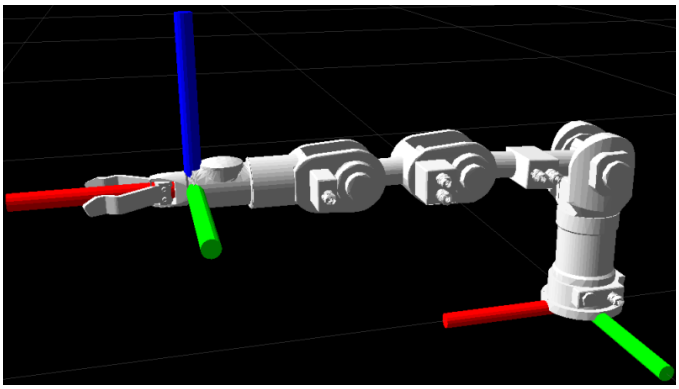


Fig. 4. The Katana 400 6m180

diameter of 1.5 cm, and place them in holes with a diameter of 1 cm. Each pick and place task was performed up to 4

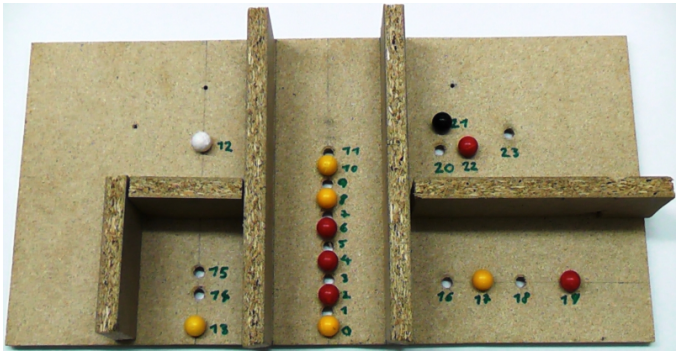


Fig. 5. Evaluation setup to perform pick and place tasks with the Katana 400 6m180

times. A task was only counted as success if the ball could be picked up or placed correctly within the hole. In order to have as many places as picks, the robot was given the ball in case of an unsuccessful pick. The test was executed with a Katana 400 6m180 robot arm. The algorithms in use are KDL and the stochastic solver. The results of this evaluation show the total success rate of pick and place tasks and can be seen in Table V. The evaluation shows the high success

Robot arm	IK solver	Pick ball	Place ball
Katana 400 6m180	KDL	0.30	0.50
Katana 400 6m180	Stochastic	0.92	0.83

TABLE V

SUCCESS RATES OF THE PICK AND PLACE TASK

of the stochastic inverse kinematic within a real environment and the impact of the inverse kinematic solver to perform a object manipulation task.

VI. CONCLUSION

In this paper we presented a novel approach to solve the inverse kinematic problem. The approach treats the inverse

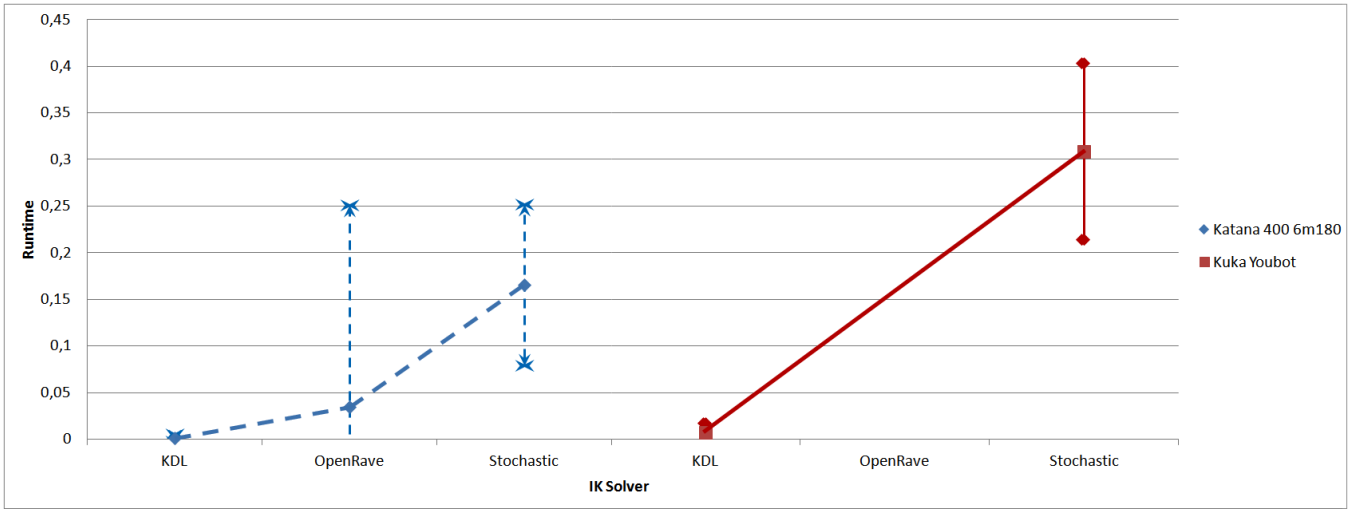


Fig. 3. Runtimes of the different inverse kinematic algorithms and their corresponding standard deviations in seconds.

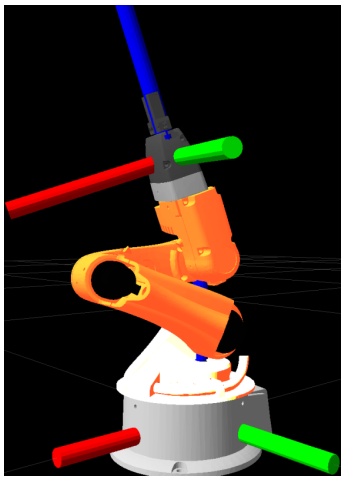


Fig. 6. The KUKA youBot

kinematic problem as optimization problem, which is solved with a global optimization algorithm. Experimental results show that the proposed inverse kinematic provides excellent success rates regarding founding a solution. In this context this novel inverse kinematic solver rules out other common inverse kinematic solvers, which are often used within the field of mobile manipulation. This advantage comes at the cost of a higher run time of the algorithm. But this drawback is acceptable in the face of a much more stable manipulation. We will address this drawback in future work by creating a fast inverse kinematic solver with a high success to find a solution by restricting the space with heuristics. Another topic of interest for future work is a evaluate how the parameters should be tuned for the algorithm.

ACKNOWLEDGMENTS

This work was supported by the Austrian Research Promotion Agency (FFG) with the ”Innovationsscheck” program.

REFERENCES

[1] M. K. Simon Bogh, Mads Hvilshoj and O. Madsen, “Autonomous industrial mobile manipulation (aimm): From research to industry,” in *Automate 2011*, 2011.

[2] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *Proceedings of IEEE Aerospace Conference*, 1999.

[3] M. C. Sachin Chitta, E. Gill Jones and K. Hsiao, “Mobille manipulation in unstructured environments,” *IEEE Robotics and automation magazine*, pp. 58–71, Jun. 2012.

[4] S. Chitta, I. Sucan, and S. Cousins, “Moveit! [ros topics],” *Robotics Automation Magazine, IEEE*, vol. 19, no. 1, pp. 18–19, March.

[5] R. Diankov, “Automated construction of robotic manipulation programs,” Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, 2010.

[6] B. B. Andrew A. Goldenberg and R. G. Fenton, “A complete generalized solution to the inverse kinematics of robots,” *IEEE Transactions on robotics and automation*, vol. RA-1, pp. 14–20, Mar. 1985.

[7] L. Kelmar and P. K. Khosla, “Automatic generation of kinematics for a reconfigurable modular manipulator system,” in *IEEE International Conference on Robotics and Automation*, Philadelphia, PA, Apr. 1988, pp. 663–668.

[8] J. K. Parker, A. R. Khoogar, and D. E. Goldberg, “Inverse kinematics of redundant robots using genetic algorithms,” in *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*. IEEE, 1989, pp. 271–276.

[9] R. Storn and K. Price, “Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces,” 1995.

Generalizing the Control Number for 6-dof UCU Hexapods with classic or eccentric U-joints

Georg Nawratil*

***Abstract*— In this paper, we present a novel index, called the Generalized Control Number ($GCTN$), which evaluates the closeness of a given non-singular configuration of a hexapod with UCU-legs to the next singularity. The $GCTN$ is invariant with respect to similarity transformations (choice of units) and under Euclidean motions of the reference frame (choice of fixed frame). Moreover, this index indicates the closeness to all types of singularities (end-effector and leg singularities) simultaneously and it has a clear geometric/kinematic meaning.**

I. INTRODUCTION

Given is a parallel manipulator with six degrees of freedom (dofs), where the fixed base is denoted by Σ_0 and the moving platform by Σ, on which the end-effector EE is installed. Moreover, Σ_0 is connected with Σ by six UCU-legs, where U denotes an universal joint and C a cylindrical one.

It is well known, that a C-joint has two dofs, where one is a translation along the cylinder axis c and the other a rotation around c. For the hexapods under consideration, only the translation along c can be controlled actively; the rotational component is passive. Therefore, the C-joint can be replaced by a composition of an active prismatic joint (P-joint) along c and a passive rotational joint (R-joint) with rotary axis c.

A U-joint also has two dofs, as it can also be seen as a serial $2R$-chain, with orthogonal axes u_1 and u_2. If these axes intersect each other, we have the classic U-joint and if this is not the case, we get a so-called *eccentric* one (cf. [3], [4]). For the UCU-legs, both types of U-joints are allowed, which are in all cases passive joints of the manipulator.

Remark 1. *According to [3], [4] the advantages of eccentric U-joints are, that they have a significantly larger pivoting range, which results in an extension of the manipulators workspace. At the same time, the joints can be produced cheaper and they can be designed more compact and stiffer, which additionally improves the accuracy.* ◇

Moreover, we assume that the connection of each U-joint with a C-joint fulfills the following two design constraints:

- the lines u_2, c and n are copunctal,
- and c intersects u_2 orthogonally,

where n denotes the common normal of u_1 and u_2 and where u_2 denotes the axis of the U-joint, which is linked with the C-joint (cf. Fig. 1, left). This assumptions keep the kinematic structure of the UCU-legs simple enough for practical application (cf. [4]).

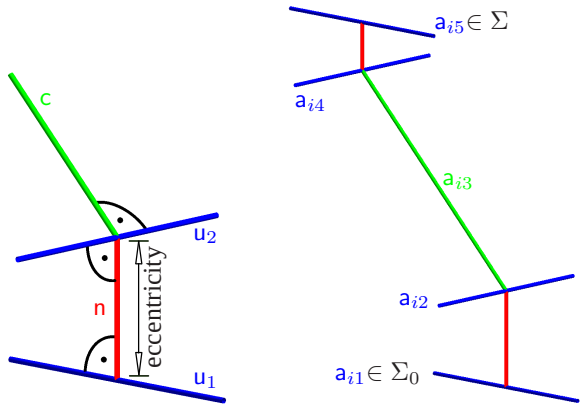


Fig. 1. Left: Connection of an eccentric U-joint and a C-joint. We get a classic U-joint, if the eccentricity equals zero. Right: Schematic sketch of the serial RRPRRR-chain, which corresponds with the i^{th} UCU-leg.

Summed up, each leg connecting Σ_0 with Σ can also be seen as a serial RRPRRR-chain, where the P-joint is active and the five R-joints are passive. We denote the j^{th} rotation axis of the i^{th} leg by a_{ij} for $i = 1, \ldots, 6$ and $j = 1, \ldots, 5$ (cf. Fig. 1, right).

Based on this notation, we first study the instantaneous kinematic of the hexapod with UCU-legs in Section II, where the different types of singularities of this manipulator are listed as well. In Section III, we make preliminary considerations on an index, which evaluates the closeness of a given non-singular configuration to the next singularity. Moreover, we discuss already existing performance indices from this point of view and repeat the so-called Control Number for Stewart Gough manipulators in more detail. Based on this, we generalize the Control Number for the hexapods under consideration in Section IV. We close the paper by demonstrating the validity of this index on the basis of a concrete example, which is given in Section V.

II. INSTANTANEOUS KINEMATICS

We use the dual vector calculus for the representation of screws and lines (cf. page 154 of [17]). Therefore, the rotation axis a_{ij} is given by

$$\underline{\mathbf{a}}_{ij} = \mathbf{a}_{ij} + \varepsilon \widehat{\mathbf{a}}_{ij}, \tag{1}$$

where $\mathbf{a}_{ij}$ is the unit vector (column vector) along the rotation axis with respect to the fixed frame. $\widehat{\mathbf{a}}_{ij}$ is the so-

*G. Nawratil is member of the Institute of Discrete Mathematics and Geometry at the Vienna University of Technology, Austria. The research was done during the author's time as interim professor at the Institute of Geometry, Technical University Dresden, Germany. Currently, the author is supported by Grant No. I 408-N13 of the Austrian Science Fund FWF within the project "Flexible polyhedra and frameworks in different spaces", an international cooperation between FWF and RFBR, the Russian Foundation for Basic Research. Email: nawratil@geometrie.tuwien.ac.at

called moment vector, which is given by $\mathbf{x}_{ij} \times \mathbf{a}_{ij}$, where $\mathbf{x}_{ij}$ is the coordinate vector (column vector) of an arbitrary point $\mathsf{X}_{ij} \in \mathsf{a}_{ij}$ with respect to the fixed frame. Further it should be noted, that ε is the dual unit, which has the property $\varepsilon^2 = 0$.

The screw for the prismatic joint of the i^{th} leg is given by

$$\underline{\mathbf{t}}_i = \mathbf{o} + \varepsilon \widehat{\mathbf{t}}_i, \tag{2}$$

where $\mathbf{o}$ denotes the zero vector and $\widehat{\mathbf{t}}_i$ the unit vector in direction of the translation with respect to the fixed frame. Therefore, in our case $\widehat{\mathbf{t}}_i$ equals $\mathbf{a}_{i3}$.

A. Jacobian matrix $\mathbf{J}_i$ of the i^{th} leg

As every leg can be seen as a serial RRPRRR-robot, the 6×6 Jacobian matrix $\mathbf{J}_i$ of the i^{th} leg can be written as (cf. [7]):

$$\mathbf{J}_i = \begin{pmatrix} \mathbf{a}_{i1} & \mathbf{a}_{i2} & \mathbf{a}_{i3} & \mathbf{a}_{i4} & \mathbf{a}_{i5} & \mathbf{o} \\ \widehat{\mathbf{a}}_{i1} & \widehat{\mathbf{a}}_{i2} & \widehat{\mathbf{a}}_{i3} & \widehat{\mathbf{a}}_{i4} & \widehat{\mathbf{a}}_{i5} & \widehat{\mathbf{t}}_i \end{pmatrix}. \tag{3}$$

Therefore, the instantaneous screw $\underline{\mathbf{q}} = \mathbf{q} + \varepsilon \widehat{\mathbf{q}}$ of Σ with respect to Σ_0 can be computed as

$$\begin{pmatrix} \mathbf{q} \\ \widehat{\mathbf{q}} \end{pmatrix} = \mathbf{J}_i \begin{pmatrix} \omega_{i1} \\ \vdots \\ \omega_{i5} \\ \tau_i \end{pmatrix}, \tag{4}$$

where ω_{ij} denotes the angular velocity of the j^{th} R-joint and τ_i the translatory velocity of the P-joint of the i^{th} leg.

The spear coordinates $(\mathbf{p}^T, \widehat{\mathbf{p}}^T)$ of the axis p (= normalized Plücker coordinates of p, cf. page 155 of [17]) of the screw $\underline{\mathbf{q}}$ can be computed according to

$$\mathbf{p} = \frac{1}{\omega}\,\mathbf{q}, \quad \widehat{\mathbf{p}} = \frac{1}{\omega}\left(\widehat{\mathbf{q}} - \frac{\omega\widehat{\omega}}{\omega^2}\,\mathbf{q}\right), \tag{5}$$

for $\omega = \|\mathbf{q}\| \neq 0$ and $\omega\widehat{\omega} = \mathbf{q}\widehat{\mathbf{q}}$. The screw parameter h is given by $h := \widehat{\omega}/\omega$, where $\widehat{\omega}$ is the translatory velocity and ω the angular velocity of the screw $\underline{\mathbf{q}}$.

If $\omega = \|\mathbf{q}\| = 0$ holds, then $\underline{\mathbf{q}}$ is an instantaneous translation along the direction $\widehat{\mathbf{q}}$. In this case, the axis is the ideal line of any plane orthogonal to $\widehat{\mathbf{q}}$.

B. Jacobian matrix $\mathbf{J}$ of the EE

If we assume that $rk(\mathbf{J}_i) = 6$ for $i = 1, \ldots, 6$, then Eq. (4) can be rewritten as

$$\mathbf{J}_i^{-1} \begin{pmatrix} \mathbf{q} \\ \widehat{\mathbf{q}} \end{pmatrix} = \begin{pmatrix} \omega_{i1} \\ \vdots \\ \omega_{i5} \\ \tau_i \end{pmatrix}. \tag{6}$$

By denoting the sixth row of $\mathbf{J}_i^{-1}$ by $(\widehat{\mathbf{j}}_i, \mathbf{j}_i)$, the 6×6 Jacobian matrix $\mathbf{J}$ of the platform can be written as

$$\mathbf{J} = \begin{pmatrix} \widehat{\mathbf{j}}_1 & \mathbf{j}_1 \\ \vdots & \vdots \\ \widehat{\mathbf{j}}_6 & \mathbf{j}_6 \end{pmatrix}. \tag{7}$$

Moreover, it should be noted that the instantaneous screw $\underline{\mathbf{j}}_i := \mathbf{j}_i^T + \varepsilon \widehat{\mathbf{j}}_i^T$ equals an instantaneous rotation around the carrier line of the i^{th} P-joint. Therefore, $(\mathbf{j}_i, \widehat{\mathbf{j}}_i)$ are the spear coordinates $(\mathbf{a}_{i3}^T, \widehat{\mathbf{a}}_{i3}^T)$ of the axis a_{i3} (cf. proof of the later given Theorem 1).

Note, that $\mathbf{J}$ transforms the instantaneous screw of the platform into the translatory velocity of the active joints, i.e.

$$\mathbf{J} \begin{pmatrix} \mathbf{q} \\ \widehat{\mathbf{q}} \end{pmatrix} = \begin{pmatrix} \tau_1 \\ \vdots \\ \tau_6 \end{pmatrix}. \tag{8}$$

C. Types of singularities

In the following, we distinguish different types of singularities:

1) $rk(\mathbf{J}_i) < 6$: This is a so-called leg singularity. Geometrically, this means that the five rotary axes and the axis of the translation (ideal line) belong to a so-called *linear line complex* (cf. Section 3 of [17]). In this case, there exist angular velocities ω_{ij} and a translatory velocity τ_i that

$$\tau_i \underline{\mathbf{t}}_i + \sum_{j=1}^{5} \omega_{ij} \underline{\mathbf{a}}_{ij} = \underline{\mathbf{o}} \tag{9}$$

holds, where $\underline{\mathbf{o}} = \mathbf{o} + \varepsilon\mathbf{o}$ denotes the zero screw. We distinguish two cases:

a) $\tau_i \neq 0$: In this case, the translatory velocity of the i^{th} active joint cannot be transmitted onto the EE, as the velocity ratio

$$(\tau_1 : \ldots : \tau_i : \ldots : \tau_6) = (0 : \ldots : 1 : \ldots : 0) \tag{10}$$

causes an instantaneous standstill of Σ; i.e. $\underline{\mathbf{q}} = \underline{\mathbf{o}}$.

b) $\tau_i = 0$: Now, there is an infinitesimal redundant mobility of the leg itself (but not of Σ). In the worst case, this can result in a self-motion of the leg.

Finally, it should be mentioned, that in a leg singularity the leg loses $6 - rk(\mathbf{J}_i)$ dofs. If an infinitesimal screw is applied to the platform, which belongs to the set of lost dofs, then this can yield a breaking of the leg.

2) $rk(\mathbf{J}) < 6$: This is a so-called EE singularity. Due to the observation of Subsection II-B, this singularity can also be interpreted by means of line geometry as follows: The hexapod is in an EE singularity, if and only if, the carrier lines of the prismatic legs belong to a linear line complex. In this case, there exists at least a screw $\underline{\mathbf{q}} \neq \underline{\mathbf{o}}$ that

$$\mathbf{J} \begin{pmatrix} \mathbf{q} \\ \widehat{\mathbf{q}} \end{pmatrix} = \begin{pmatrix} \mathbf{o} \\ \mathbf{o} \end{pmatrix} \tag{11}$$

holds. Therefore, in an EE singularity, the platform is infinitesimal movable while all active joints are locked. Finally it should be noted, that in the worst case, this singularity can result in a self-motion of Σ.

Remark 2. *This singularity study also shows, that the hexapods under consideration only have line-based singularities, even though the last three joints of each leg are not equivalent with a spherical joint (S-joint), if an eccentric U-joint is used at Σ. Therefore, these are more general parallel manipulators with line-based singularities, than those characterized in Section 4 of [2].* $\diamond$

III. PERFORMANCE INDEX

In future applications, it is planned that the hexapod's motion is controlled directly by ordinary skilled workers and not by highly-qualified academics, e.g. wheel loaders will be coupled by hexapods with different EEs (dredger bucket, stacker forks, snowplough, gripper, ...).[1] Therefore, there is an interest in an index, which gives the operator a feedback about the closeness of a given non-singular hexapod-configuration to the next singular one.

As it is well known, that there does not exist a distance metric in the pure mathematical sense, if rotational and translatory dofs are involved (which is the case for a 6-dof hexapod), we are looking for a performance index PI, which assigns to each configuration $\mathcal{C}$ a scalar $PI(\mathcal{C}) \in \mathbb{R}$ obeying the following six properties:

1) $PI(\mathcal{C}) \geq 0$ for all $\mathcal{C}$ of the configuration space,
2) $PI(\mathcal{C}) = 0$ if and only if $\mathcal{C}$ is singular,
3) $PI(\mathcal{C})$ is invariant under Euclidean motions of the reference frame,
4) $PI(\mathcal{C})$ is invariant under similarities,
5) $PI(\mathcal{C})$ has a geometric/kinematic meaning,
6) $PI(\mathcal{C})$ is computable in real-time.

A further challenge for the definition of the requested index is, that it has to evaluate the closeness to different types of singularities simultaneously, as separated computations of the closeness to EE singularities and leg singularities (for each leg) go at the expense of the computation time (cf. demand 6), and one is confronted with the problem of combining the obtained values to a single meaningful closeness index (cf. demand 5). But exactly this clear geometric/kinematic meaning is of importance for identifying a critical value, which indicates that a given configuration is too close to a singularity for guaranteeing a save performance of the hexapod.

As the set of singular poses of a manipulator is solely determined by its geometry, a performance index, which makes demands to evaluate the closeness to the next singularity, should only depend on geometric/kinematic properties of the inspected non-singular pose. Therefore, such a performance index must not depend on the EE. As a consequence, all known condition number indices (either based on the characteristic point [21], operation ellipsoid [14], [15] or velocity of three EE points [8]) as well as the local singularity transmission index [11], which depends on the choice of the application point, are out of question.

Moreover, the requested index must not depend on non-kinematic parameters as mass or stiffness, which exclude also the indices presented in [1], [6], [18]. In the following we discuss the remaining EE independent performance indices, which are known to the author, in more detail:

[1]Cf. research project *"MOBIMA – Arbeitsausrüstungen mit parallelkinematischen Strukturen für mobile Arbeitsmaschinen"* funded by the German Ministry of Education and Research. For more details please see: `http://tu-dresden.de/die_tu_dresden/fakultaeten/fakultaet_maschinenwesen/iwm/forschung/2012_mobima`

A. Manipulability [20]

A drawback of the manipulability $M(\mathcal{C})$ is, that it is not invariant under similarity transformations and therefore it depends on the choice of units (cf. [12]). To overcome this problem, some authors use the following relation as index:

$$M^\star := \frac{M(\mathcal{C})}{M(\mathcal{C}_{max})}, \tag{12}$$

with $\mathcal{C}_{max}$ denoting the configuration of the hexapod, where the manipulability is maximal ($\Rightarrow M^\star \in [0,1]$). But the computation of $\mathcal{C}_{max}$ is a highly non-linear task and was only done for some special manipulators of Stewart Gough (SG) type[2] (cf. [9], [10]). Moreover, only in some special cases $M(\mathcal{C}_{max})$ can be interpreted geometrically as the volume, spanned by the framework (cf. [9]).

B. Best fitting linear line complex [16], [19]

As our studied manipulator only has line-based singularities (cf. Remark 2), also this index has to be taken into consideration, which is again not invariant under similarities. But one can solve this problem as for the case of the manipulability.

The much bigger problem is, that this index does not consider singular linear line complexes, where the axis is an ideal line (cf. page 166 of [17]). In order to close this gap, the authors of [16] proposed the computation of a second index. But it is not clear how these two indices should be combined to a single geometric/kinematic meaningful value, evaluating the closeness to the next linear line complex.

Beside the already mentioned drawbacks of the manipulability and the method of the best fitting linear line complex, these two indices cannot master the challenge formulated in the paragraph below the six demands.

C. Control Number [13], [14], [15]

As pointed out by the author in [13], [14], [15], the Control Number CTN fulfills all six demands. Therefore, this index is best suited for measuring the closeness to the next singularity in the author's mind, but until now the CTN is only defined for SG manipulators. As we want to generalize the CTN for the hexapods under consideration within the next section, we repeat its basic idea and definition in the following two paragraphs:

As in each pose, the SPS-leg allows a rotational self-motion around the line spanned by the centers of the S-joints, we are only interested in an index, which evaluates the closeness to EE singularities. Note that EE singularities of SG manipulators have the same geometric interpretation as the one given in item 2 of Subsection II-C. Therefore, SG manipulators are also infinitesimal movable in EE singularities, which means that there exists an infinitesimal motion of Σ while all actuators are locked. As a consequence, the velocity of Σ can be arbitrarily large (even infinity), and therefore the posture is uncontrollable. In practice,

[2]These are hexapods with six SPS-legs, where both S-joints are passive and the P-joint is active.

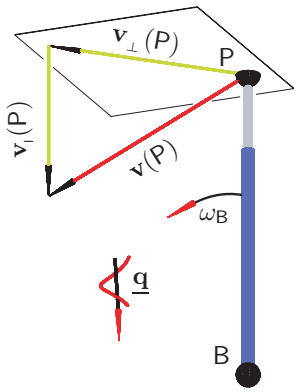


Fig. 2. Definition of the angular velocity ω_B of the spherical base joint (with center B): $\mathbf{v}(P)$ denotes the velocity of the platform point P (= center of S-joint) with respect to the instantaneous screw $\underline{\mathbf{q}}$. Moreover, $\mathbf{v}_\parallel(P)$ (resp. $\mathbf{v}_\perp(P)$) is the component of $\mathbf{v}(P)$ along (resp. orthogonal to) the carrier line of the P-joint. Then, ω_B is defined as the length of $\mathbf{v}_\perp(P)$ divided through the leg length. Note, that the definition of ω_P can be done analogously with respect to the inverse motion $-\underline{\mathbf{q}}$.

configurations must be avoided, where minor (or even zero) variations of the leg lengths have uncontrollable large effects on the instantaneous displacement of Σ. The question is, which measurable parameter of the SG manipulator indicates the circumstance of uncontrollability in a natural way and has a geometric/kinematic meaning for the manipulator.

The answer to this question are the angular velocities of the S-joints (cf. Fig. 2). We computed the maximum λ_{max} and the minimum λ_{min} of the sum of the squared angular velocities of the passive joints under the normalizing condition that the sum of the squared translatory velocities of the active joints equals 1. Then, the CTN is defined as:

$$CTN := \sqrt{\frac{\lambda_{min}}{\lambda_{max}}} \in [0,1]. \tag{13}$$

Remark 3. *For a more detailed review of the indices discussed in Subsection III-A, III-B and III-C for parallel manipulators of SG type, please see [15]. Moreover, these indices are also compared within Subsection 6.4 of [15].* $\diamond$

IV. GENERALIZED CONTROL NUMBER

The base for the definition of a generalized version of the CTN is the following theorem:

Theorem 1. *A leg singularity of type (a) with $rk(\mathbf{J}_i) = 5$ cannot exist.*

Proof: As all rotary axes $\mathsf{a}_{i1}, \ldots, \mathsf{a}_{i5}$ intersect (or are even identical with) the carrier line of the i^{th} P-joint (= line a_{i3}), the linear line complex spanned by the axes $\mathsf{a}_{i1}, \ldots, \mathsf{a}_{i5}$ equals the path normal complex of the instantaneous screw $\underline{\mathbf{j}}_i$ (cf. page 164 of [17]). This linear line complex is a singular one and uniquely determined, if $\underline{\mathbf{a}}_{i1}, \ldots, \underline{\mathbf{a}}_{i5}$ are linearly independent ($\Rightarrow rk(\mathbf{J}_i) = 5$).

Therefore, a leg singularity of type (a) with $rk(\mathbf{J}_i) = 5$ exists, if and only if, the axis t of $\underline{\mathbf{t}}_i$ intersects the carrier line of the P-joint. But this can never happen, as t is the ideal line of the plane orthogonal to a_{i3}. $\square$

A consequence of this theorem is, that a leg singularity of type (a) can only occur if $rk(\mathbf{J}_i) < 5$ holds, but this implies the existence of a leg singularity of type (b). Therefore, our performance index only has to indicate EE singularities and leg singularities of type (b). The common characteristic property of these two singularities is, that there exists an infinitesimal mobility while all active joints are fixed. Therefore, the so-called Generalized Control Number $GCTN$ can be used as index. The definition and computation of the $GCTN$ is given as follows:

We calculate the extreme values of the objective function (sum of the squared angular velocities of the passive joints)

$$\zeta: \sum_{i=1}^{6}\sum_{j=1}^{5}\omega_{ij}^2 \tag{14}$$

under the normalizing condition (sum of the squared translatory velocities of the active joints)

$$\nu: \sum_{i=1}^{6}\tau_i^2 - 1 = 0. \tag{15}$$

Under consideration of Eq. (6), these two equations can be expressed in dependency of $\underline{\mathbf{q}}$. As the resulting equations are quadratic functions in $\underline{\mathbf{q}}$, they can be written as:

$$\zeta(\underline{\mathbf{q}}): \ (\mathbf{q}^T, \widehat{\mathbf{q}}^T)\mathbf{Z}\begin{pmatrix}\mathbf{q}\\ \widehat{\mathbf{q}}\end{pmatrix}, \tag{16}$$

and

$$\nu(\underline{\mathbf{q}}): \ (\mathbf{q}^T, \widehat{\mathbf{q}}^T)\mathbf{N}\begin{pmatrix}\mathbf{q}\\ \widehat{\mathbf{q}}\end{pmatrix} - 1 = 0, \tag{17}$$

where $\mathbf{Z}$ and $\mathbf{N}$ are 6×6 matrices, with $\mathbf{N} = \mathbf{J}^T\mathbf{J}$.

We solve the optimization problem by introducing a Lagrange multiplier λ (cf. [5]). Then, the approach simplifies under consideration of

$$\nabla\zeta(\underline{\mathbf{q}}) = 2\,\mathbf{Z}\begin{pmatrix}\mathbf{q}\\ \widehat{\mathbf{q}}\end{pmatrix}, \quad \nabla\nu(\underline{\mathbf{q}}) = 2\,\mathbf{N}\begin{pmatrix}\mathbf{q}\\ \widehat{\mathbf{q}}\end{pmatrix}, \tag{18}$$

to the general eigenvalue problem

$$(\mathbf{Z} - \lambda\,\mathbf{N})\begin{pmatrix}\mathbf{q}\\ \widehat{\mathbf{q}}\end{pmatrix} = \begin{pmatrix}\mathbf{o}\\ \mathbf{o}\end{pmatrix}. \tag{19}$$

This system of linear equations only has a non-trivial solution, if the determinant of the matrix $\mathbf{Z} - \lambda\,\mathbf{N}$ vanishes. Each solution λ_i (general eigenvalue of $\mathbf{Z}$ with respect to $\mathbf{N}$) of the resulting characteristic polynomial of degree 6 in λ corresponds with a general eigenvector $\mathbf{e}_i$. Due to Eqs. (17) and (19) we get

$$\zeta(\mathbf{e}_i) = \lambda_i, \tag{20}$$

which implies that the greatest λ_+ and smallest λ_- general eigenvalue equal the requested extrema.

Theorem 2. *The $GCTN$, which is given by*

$$GCTN := \sqrt{\frac{\lambda_-}{\lambda_+}} \in [0,1], \tag{21}$$

fulfills all six stated requirements.

Proof: Due to the definition of the index, all demands, with exception of the second one, are trivially fulfilled. Therefore, we only comment on demand 2: The value of λ_+ equals ∞, if and only if, the manipulator is in an EE singularity or leg singularity of type (b), as only in these configurations an instantaneous self-mobility of the manipulator exists, while all active actuators are locked (cf. Section II-C).

Hence, it remains to check the case $\lambda_- = 0$: In this case, all passive joints have an instantaneous standstill. As a consequence, an instantaneous change of the EE's orientation is not possible and therefore only a pure translation can be performed at this moment. A pure translation can only be done if all six legs are parallel to each other, but this already implies $rk(\mathbf{J}) \leq 3$, as the six carrier lines of the P-joints belong to a bundle of lines (cf. page 142 of [17]). □

Remark 4. *Note, that the $GCTN$ also masters the challenge formulated in the paragraph below the six demands. It should be mentioned that, similar to the CTN (cf. [14]), the $GCTN$ can also be used for parallel manipulators with more than six legs, i.e. redundant hexapods with UCU-legs.* ◇

If a given configuration of the hexapod is indicated by a small $GCTN$-value to be close to a singularity, then the corresponding eigenvector $\mathbf{e}_+$ of λ_+ contains the following extra information:

The joint ratio $\mathbf{r}^+ := (\tau_1^+ : \ldots : \tau_6^+)$, computed from $\underline{\mathbf{q}}_+ := \mathbf{q}_+ + \varepsilon\widehat{\mathbf{q}}_+$ with $(\mathbf{q}_+^T, \widehat{\mathbf{q}}_+^T) := \mathbf{e}_+^T$ by Eq. (8), corresponds with the most uncontrollable motion of the hexapod, as small variations of the prismatic joints have large effects on the instantaneous transformation of the whole manipulator (EE singularity) or of a substructure (leg singularity). Therefore, the joint ratio $\mathbf{r}^+$ should be avoided.

This joint ratio $\mathbf{r}^+$ can also be used for evaluating the quality of an arbitrary instantaneous joint ratio $\mathbf{r} := (\tau_1 : \ldots : \tau_6)$ by computing the angle ρ enclosed by the one-dimensional subspaces $\mathbf{r}$ and $\mathbf{r}^+$:

$$\rho := \arccos \frac{\pm \mathbf{r} \cdot \mathbf{r}^+}{\|\mathbf{r}\| \|\mathbf{r}^+\|}, \tag{22}$$

where the sign $\pm$ has to be chosen that $\rho \in [0, \pi/2]$ holds.

Moreover, we can even detect whether the given configuration is close to either an EE singularity or a leg singularity by computing

$$\mu_i(\underline{\mathbf{q}}_+) := \sum_{j=1}^{5} \omega_{ij}^2 \tag{23}$$

for $i = 1, \ldots, 6$. If $\mu_i(\underline{\mathbf{q}}_+)$ is not far away from λ_+, then the manipulator is in the neighborhood of a leg singularity of the i^{th} leg, as $\mu_1(\underline{\mathbf{q}}_+) + \ldots + \mu_6(\underline{\mathbf{q}}_+) = \lambda_+$ holds. Otherwise, we are close to an EE singularity.

Remark 5. *Note, that the $GCTN$ can also be used to optimize the kinematic design of the hexapods under consideration, as this was done for SG manipulators with respect to the CTN in [14], [15]. From this perspective, the hexapod should be isotropic in the central configuration $\mathcal{C}_\odot$ of the workspace, i.e. $GCTN(\mathcal{C}_\odot) = 1$. The topic of isotropy is dedicated to future research.* ◇

V. EXAMPLE

Due to the simplicity of the inverse kinematics, we study a hexapod, where both U-joints of all UCU-legs are classic ones. Moreover, we assume that the centers B_i and P_i of the base U-joint and platform U-joint, respectively, of the i^{th} leg are located on semi-regular hexagons with a circumcircle of radius 1. Without loss of generality, we can choose a Cartesian coordinate system in Σ_0, that B_i and P_i have the following coordinate vectors $\mathbf{b}_i$ and $\mathbf{p}_i$, respectively:

$$\mathbf{b}_i = (\cos\alpha_i, \sin\alpha_i, 0)^T, \quad \mathbf{p}_i = (\cos\beta_i, \sin\beta_i, d)^T,$$

with

$$\begin{aligned}
\alpha_1 &= \beta_2 - \frac{\pi}{3} = -\alpha, & \alpha_2 &= \beta_1 + \frac{\pi}{3} = \alpha,\\
\alpha_3 &= \beta_4 - \frac{\pi}{3} = \frac{2\pi}{3} - \alpha, & \alpha_4 &= \beta_3 + \frac{\pi}{3} = \frac{2\pi}{3} + \alpha,\\
\alpha_5 &= \beta_6 - \frac{\pi}{3} = \frac{4\pi}{3} - \alpha, & \alpha_6 &= \beta_5 + \frac{\pi}{3} = \frac{4\pi}{3} + \alpha.
\end{aligned}$$

Moreover, we set the design parameter α equal to $\pi/12$ and the configuration parameter d equal to 1 in order to get a presentable graphical illustration of the hexapod's central configuration $\mathcal{C}_\odot$. In addition, we can still select the direction of the first rotational axis a_{i1} through B_i as well as the direction of the last rotational axis a_{i5} through P_i. They are chosen in a way, that they contain the center of the corresponding circumcircle (cf. Fig. 3).

By rotating the platform of $\mathcal{C}_\odot$ around the line g spanned by the centers of the two circumcircles about the angle $\pi/2$, we get into the EE singularity illustrated in Fig. 4. The value of the $GCTN$, in dependency of the rotation angle $\delta \in [0, \pi/2]$, is displayed in Fig. 6.

By rotating the platform of $\mathcal{C}_\odot$ around g about the angle $\pi/6$, we get into the intermediate pose $\mathcal{C}_\oslash$, where the axis a_{13} is parallel to g. If we move the hexapod out of $\mathcal{C}_\oslash$, by rotating the platform about the angle $\pi/4$ around the line h, which passes through the center of the platform and is orthogonal to the plane spanned by a_{13} and g, then we get into the leg singularity illustrated in Fig. 5. The value of the $GCTN$, in dependency of the rotation angle $\theta \in [0, \pi/4]$ of the rotation around h, is displayed in Fig. 6.

In the following we study a configuration close to the leg singularity and EE singularity, respectively, from the perspective of the last paragraph before Remark 5. For e.g. $\theta = 40°$, we get $GCTN \approx 0.052$, $\lambda_+ \approx 630.867$ and

$$\begin{aligned}
&\mu_1 \approx 600.997, && \mu_2 \approx 7.271, && \mu_3 \approx 4.472,\\
&\mu_4 \approx 3.370, && \mu_5 \approx 3.806, && \mu_6 \approx 10.951,
\end{aligned}$$

which shows that we are close to a leg singularity of the first leg. In contrast, for the configuration given by $\delta = 85°$, we get $GCTN \approx 0.034$, $\lambda_+ \approx 2267.560$ and

$$\mu_1 = \mu_3 = \mu_5 \approx 508.978, \quad \mu_2 = \mu_4 = \mu_6 \approx 246.875.$$

According to the prognosticate behaviour, we are close to an EE singularity.

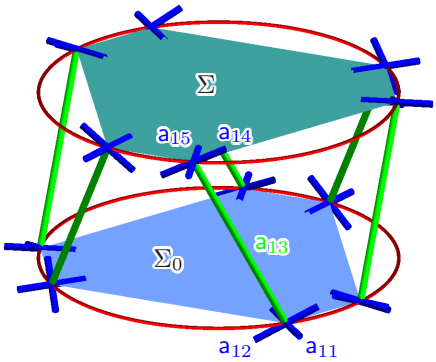


Fig. 3. The hexapod in its central configuration $\mathcal{C}_\odot$. The manipulator is far from being isotropic as $GCTN(\mathcal{C}_\odot) \approx 0.314$ holds. E.g. the corresponding octahedral manipulator ($\alpha = 0$) in this pose has a $GCTN$ of about 0.715.

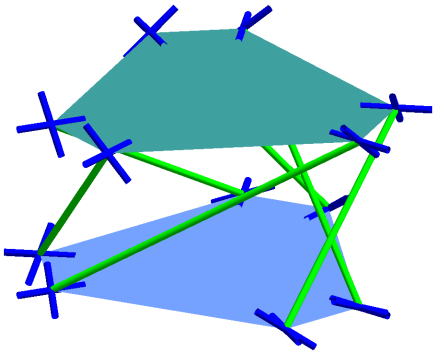


Fig. 4. A well-known EE singularity of the hexapod. Note, that the camera position for the Figs. 3, 4 and 5 is always the same with respect to Σ_0.

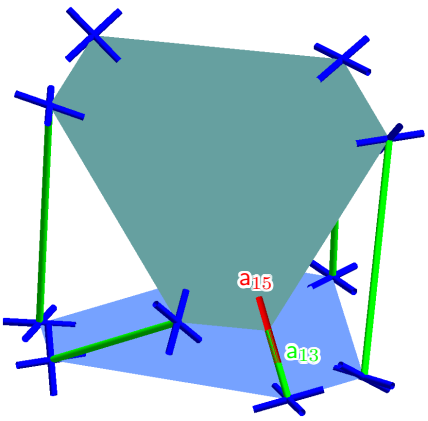


Fig. 5. Leg singularity of the hexapod: The axes a_{13} (green) and a_{15} (red) coincide. The axis a_{14} is not displayed, as it is not uniquely determined due to a rotational self-mobility of the leg around the axis $\mathsf{a}_{13} = \mathsf{a}_{15}$.

ACKNOWLEDGMENT

The author wants to thank Bernd Kauschinger and his student Felix Bender from the Institute of Machine Tools and Control Technology at the Technical University Dresden, Germany, for bringing the author's attention to the topic dealt within this paper, and for the fruitful discussions in this context during the author's stay in Dresden (cf. Footnote *).

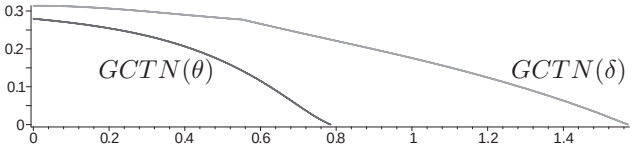


Fig. 6. The $GCTN$-graphs in dependency of the rotation angles δ and θ.

REFERENCES

[1] BIER, C.C.: *Geometrische und physikalische Analyse von Singularitäten bei Parallelstrukturen.* Doctoral thesis. Institute of Technology, University of Braunschweig (2006)

[2] FANGLI, H., MCCARTHY, J.M.: *Conditions for Line-Based Singularities in Spatial Platform Manipulators.* Journal of Robotic Systems 15(1) 43–55 (1998)

[3] GROSSMANN, K., KAUSCHINGER, B., RIEDEL, M.: *Exzentrische Gelenke für parallelkinematische Werkzeugmaschinen.* Zeitschrift für wirtschaftlichen Fabrikbetrieb 107(1–2) 25–32 (2012)

[4] GROSSMANN, K., KAUSCHINGER, B.: *Eccentric universal joints for parallel kinematic machine tools: variants and kinematic transformations.* Production Engineering - Research and Development 6(4–5) 521–529 (2012)

[5] HESTENES, M.R.: *Optimization theory.* Wiley publication (1975)

[6] HUBERT, J., MERLET, J.-P.: *Static of Parallel Manipulators and Closeness to Singularity.* ASME Journal of Mechanisms and Robotics 1(1) 011011 (2009)

[7] HUSTY, M., KARGER, A., SACHS, H., STEINHILPER, W.: *Kinematik und Robotik.* Springer (1997)

[8] KIM, S.-G., RYU, J.: *New Dimensionally Homogeneous Jacobian Matrix Formulation by Three End-Effector Points for Optimal Design of Parallel Manipulators.* IEEE Transactions on Robotics and Automation 19(4) 731–737 (2003)

[9] LEE, J., DUFFY, J., HUNT, H.: *A Practiacl Quality Index Based on the Octahedral Manipulator.* International Journal of Robotics Research 17(10) 1081–1090 (1998)

[10] LEE, J., DUFFY, J.: *The optimum quality index for some spatial inparallel devices.* Florida conference on Recent Advances in Robotics, Gainesville, USA (1999)

[11] LIU, X.-J., WU, C., WANG J.: *A new approach for singularity analysis and closeness measurement to singularities of parallel manipulators.* ASME Journal of Mechanisms and Robotics 4(4) 041001 (2012)

[12] MERLET, J.-P.: *Jacobian, Manipulability, Condition Number, and Accuracy of Parallel Robots.* ASME Journal of Mechanical Design 128 199–206 (2006)

[13] NAWRATIL, G.: *The Control Number as Index for Stewart Gough Platforms.* Advances in Robot Kinematics: Mechanisms and Motion (J. Lenarcic, B. Roth eds.), 15–22, Springer (2006)

[14] NAWRATIL, G.: *New Performance Indices for 6-dof UPS and 3-dof RPR Parallel Manipulators.* Mechanism and Machine Theory 44(1) 208–221 (2009)

[15] NAWRATIL, G.: *Neue kinematische Performance Indizes für 6R Roboter und Stewart Gough Plattformen.* Doctoral thesis. Institute of Discrete Mathematics and Geometry, Vienna University of Technology (2007)

[16] POTTMANN, H., PETERNELL, M., RAVANI, B.: *Approximation in line space - applications in robot kinematics and surface reconstruction.* Advances in Robot Kinematics: Analysis and Control (J. Lenarcic, M. Husty eds.), 403–412, Kluwer (1998)

[17] POTTMANN, H., WALLNER, J.: *Computational Line Geometry.* Springer (2001)

[18] VOGLEWEDE, P.A., EBERT-UPHOFF, I.: *Overarching Framework for Measuring Closeness to Singularities of Parallel Manipulators.* IEEE Transactions on Robotics 21(6) 1037–1045 (2005)

[19] WOLF, A., SHOHAM, M.: *Investigations of Parallel Manipulators Using Linear Complex Approximation.* Journal of Mechanical Design 125 564–572 (2003)

[20] YOSHIKAWA, T.: *Manipulability of Robotic Mechanisms.* Int. Journal of Robotics Research 4(2) 3–9 (1985)

[21] ZANGANEH, K.E., ANGELES, J.: *Kinematic Isotropy and the Optimum Design of Parallel Manipulators.* Int. Journal of Robotics Research 16(2) 185–197 (1997)

A Time Optimal Solution for the Waiter Motion Problem with an Industrial Robot

Matthias Oberherber, Hubert Gattringer, and Klemens Springer

***Abstract*— This paper presents a time optimal solution for the "Waiter-Motion-Problem". The goal is to move a cup, that is loosely placed at the end-effector of an industrial robot, as fast as possible on a specified path. The path is defined with splines represented by Bernstein polynomials. By introducing a path parameter, the optimization problem is transformed into the phase plane and solved with a Bellman optimization strategy. Physical limitations like motor velocities, motor torques and friction forces are considered. The cycle time is reduced even more by adapting the orientation of the tray. Simulation as well as experimental results are presented.**

I. INTRODUCTION

Time optimal path planning is an interesting topic for industrial use. The advantage is to save cycle time by exploiting the robots mechanical and electrical limits. There exist a lot of papers concerning this subject. Especially the works of Pfeiffer et al. [5], Shin et al. [6] and Bobrow et al. [8] provide the basis for effective algorithms. A special challenge represents the so called "Waiter-Motion-Problem" - a robot with a tray mounted on the end-effector should follow a given spatial path in shortest possible time without bringing an object placed on the plate to slip. Geu Flores et al. discussed this problem in [1]. They use quintic B-splines to define the spatial path whereby the time optimal solution is obtained with an special algorithm that is based on forward and backward integration of the equations of motion to find optimal switching points. In contrast to that, in our paper the interpolation between predefined points is done with Bernstein polynomials. Additionally, the time optimal solution is calculated with the help of the Bellman strategy, simply because of greater flexibility in relation to path definitions and fast solution times for the optimization algorithm.

The paper is organized as follows: In section II general definitions for the description of the spatial path are given, while the equations of motions for the robot are calculated in section III. The general time optimization of this problem under specific physical limitations is shown in IV. Additional constraints for the "Waiter-Motion-Problem", namely acceleration constraints of the end-effector, are included in section V. The overall cycle time is reduced even more by optimizing the end-effector orientation (section VI). Finally in section (VII) a comparison between the optimization with constant and optimized orientation of the tray can be found.

Johannes Kepler University Linz
Altenbergerstr. 69, 4040 Linz, Austria
m.oberherber@gmx.at, hubert.gattringer@jku.at, klemens.springer@jku.at
http://www.robotik.jku.at

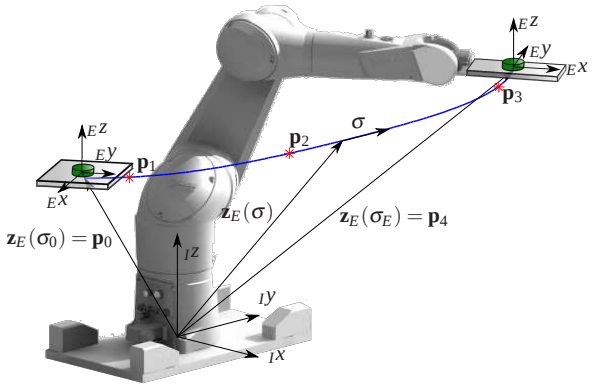


Fig. 1. Path planning by interpolation of points

The experimental verification is done with a Stäubli Rx130 industrial robot.

II. SPATIAL PATH PLANNING

A path $\mathbf{z}_E = [x_E, y_E, z_E]^T$ for the end-effector in the inertial coordinate system is calculated by interpolating pre-defined points $\mathbf{p}_j$ with splines within the robots working space, see Fig. 1. This path can be parametrized with a (scalar) path parameter σ

$$\mathbf{z}_E = \mathbf{z}_E(\sigma). \tag{1}$$

The time behavior $\sigma(t)$ has to be optimized between the range $\sigma_0 = 0$ (start point) and $\sigma_E = 1$ (end point). This parametrization is utilized with Bernstein polynomials, in the following way

$$\begin{aligned} &R_j^d(\sigma) = 0 \text{ for } \quad j = 0,\ldots,n-d-1 \\ &R_{n-d+k}^d(\sigma) = \begin{pmatrix} d \\ k \end{pmatrix} \frac{(\sigma-\sigma_0)^k(\sigma_E-\sigma)^{d-k}}{(\sigma_E-\sigma_0)^d} \text{ for } \quad k = 0,\ldots,d \\ &R_j^d(\sigma) = 0 \text{ for } \quad j = n+1,\ldots,2n-d \ , \end{aligned} \tag{2}$$

see Fig. 2 for a graphical representation with a polynomial degree of $d = 4$. Parameter n is the maximum polynomial degree depending on the number of interpolation points.
In contrast to the spatial positions of the nodes $\mathbf{p}_j$, the discretization of σ is not unique. Especially two methods for the parametrization are available:

- Equidistant parametrization:
  The m nodes are equidistantly distributed over the range of the curve parameter $\sigma_0 \ldots \sigma_E$
  $\sigma_k = k\Delta\sigma$, $k = 1..m-1$, with $\Delta\sigma = \frac{\sigma_E-\sigma_0}{m-1}$.

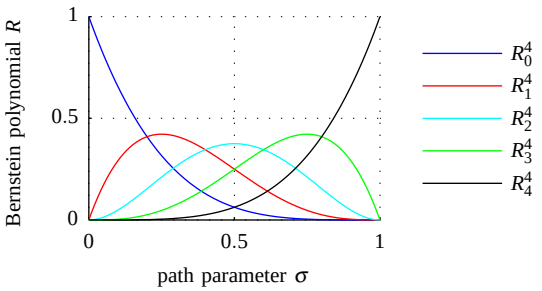


Fig. 2. Bernstein polynomial of degree $d = 4$

- Chordal parametrization:
  The nodes are distributed over the range of the path parameter $\sigma_0 \dots \sigma_E$ under consideration of the distance of the defined points in space. For that, the length of the polygonial-line is calculated as

$$L = \sum_{j=0}^{m-n-3} \left\| \mathbf{p}_{j+1} - \mathbf{p}_j \right\|$$

  and furthermore the discretization points are defined by

$$\sigma_k = \sigma_0 + \frac{\sigma_E - \sigma_0}{L} \sum_{j=0}^{k-1} \left\| \mathbf{p}_{j+1} - \mathbf{p}_j \right\|.$$

With this chordal parametrization on hand, the end-effector path can be written as

$$\mathbf{z}_E(\sigma) = \sum_{j=0}^{m-n-2} \mathbf{d}_j R_j^d(\sigma), \tag{3}$$

where the spline nodes

$$\mathbf{D} = (\mathbf{d}_0, \mathbf{d}_1 \dots \mathbf{d}_{m-n-2}) \tag{4}$$

are the result of the linear equation

$$\mathbf{R}_B \mathbf{D} = \mathbf{P}. \tag{5}$$

Beside the Bernstein polynomials

$$\mathbf{R}_B = \begin{bmatrix} R_0^d(\sigma_0) & \dots & R_{m-n-2}^d(\sigma_0) \\ R_0^d(\sigma_1) & \dots & R_{m-n-2}^d(\sigma_1) \\ \vdots & \ddots & \vdots \\ R_0^d(\sigma_{m-n-2}) & \dots & R_{m-n-2}^d(\sigma_{m-n-2}) \end{bmatrix} \tag{6}$$

this equation also contains the pre-defined path points in space $\mathbf{P} = (\mathbf{p}_0, \mathbf{p}_1 \dots \mathbf{p}_{m-n-2})$. Since $\mathbf{R}_B$ has band structure, the inversion can be easily performed. More details on splines can be found in [2] and [3].

III. DYNAMIC MODELING

The equations of motion have to be calculated on the one hand to be able to simulate the robotic system and on the other hand to calculate a model-based feed-forward control that has to be used for the control of the real robot and also for the time optimization. We use the Projection Equation

$$\sum_{i=1}^{N} \left(\left(\frac{\partial_R \mathbf{v}_c}{\partial \dot{\mathbf{q}}} \right)^T \quad \left(\frac{\partial_R \omega_c}{\partial \dot{\mathbf{q}}} \right)^T \right) \begin{pmatrix} {}_R\dot{\mathbf{p}} + {}_R\tilde{\omega}_{IR}\, {}_R\mathbf{p} - {}_R\mathbf{f}^e \\ {}_R\dot{\mathbf{L}} + {}_R\tilde{\omega}_{IR}\, {}_R\mathbf{L} - {}_R\mathbf{M}^e \end{pmatrix}_i = 0. \tag{7}$$

see [6] for details, since it is a powerful method for multi-body systems to calculate the equations of motion. Linear momenta $\mathbf{p} = m\ \mathbf{v}_c$ and angular momenta $\mathbf{L} = \mathbf{J}\ \omega_c$ are projected into the minimal space (minimal velocities $\dot{\mathbf{q}}$) via the appropriate Jacobian matrices. All the values like the translational velocity $\mathbf{v}_c$ or the rotational velocity of the center of gravity ω_c can be inserted in arbitrary coordinate systems R. In contrast to ω_c, ω_{IR} is the velocity of the used reference system. The matrix $\mathbf{J}$ is the inertial tensor, while $\tilde{\omega}\ \mathbf{p}$ characterizes the vector product $\omega \times \mathbf{p}$. $\mathbf{f^e}$ and $\mathbf{M^e}$ are imposed forces and moments acting on the ith body. Evaluating the Projection Equation for $N = 12$ bodies (the industrial robots consists of six motors and six links) leads to the equations of motion

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{g}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{Q}. \tag{8}$$

where $\mathbf{M}$ is the position dependent, positive definite, symmetric mass matrix, $\mathbf{g}$ contains all nonlinear terms like gravitational-, Coriolis- , centrifugal and friction forces, while $\mathbf{Q}$ are the motor torques. Since the robot is fully actuated, the feed-forward torques $\mathbf{Q}_d$ can directly be calculated via inverse dynamics of (8) by inserting desired values (subscript d) for the path

$$\mathbf{Q}_d = \boldsymbol{f}(\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d) = \mathbf{M}(\mathbf{q}_d)\ddot{\mathbf{q}}_d + \mathbf{g}(\mathbf{q}_d, \dot{\mathbf{q}}_d). \tag{9}$$

This feed-forward torques are used to include actuator torque constraints in the next section.

IV. TIME OPTIMAL MOTION GENERATION

A. Parametrization of the equation of motion

The geometric path $\mathbf{z}_E$ (3) is defined in world coordinates and parametrized by the path parameter σ. However, physical constraints like joint velocities, accelerations and torques

$$\begin{aligned} \dot{\mathbf{q}}_{min} \le \dot{\mathbf{q}} \le \dot{\mathbf{q}}_{max} \\ \ddot{\mathbf{q}}_{min} \le \ddot{\mathbf{q}} \le \ddot{\mathbf{q}}_{max} \\ \mathbf{Q}_{min} \le \mathbf{Q} \le \mathbf{Q}_{max} \end{aligned} \tag{10}$$

are defined in joint coordinates. Therefore a transformation, the inverse kinematics

$$\mathbf{q} = \mathbf{f}(\mathbf{z}_E(\sigma)) \tag{11}$$

which can be solved analytically, has to be performed. Additionally, the joint velocities

$$\dot{\mathbf{q}} = \frac{\partial \mathbf{q}}{\partial \sigma} \dot{\sigma} = \mathbf{q}' \dot{\sigma} \tag{12}$$

and the joint accelerations

$$\ddot{\mathbf{q}} = \frac{\partial}{\partial \sigma}(\mathbf{q}'\dot{\sigma})\dot{\sigma} = \mathbf{q}''\dot{\sigma}^2 + \frac{1}{2}\mathbf{q}'(\dot{\sigma}^2)' \tag{13}$$

can directly be computed with respect to the path parameter and its derivatives. Substituting (11,12,13) into (9) yields the equations of motion with respect to the path parameter

$$\mathbf{Q} = \mathbf{A}(\sigma)(\dot{\sigma}^2)' + \mathbf{B}(\sigma)(\dot{\sigma}^2) + \mathbf{C}(\sigma) + \mathbf{D}(\sigma)\dot{\sigma} \tag{14}$$

(subscript d omitted). The corresponding quantities $\mathbf{A},\mathbf{B},\mathbf{C}$ can e.g. be determined by

$$\mathbf{A} = \frac{1}{2} f(\ddot{\mathbf{q}} = \mathbf{q}', \dot{\mathbf{q}} = 0, \mathbf{q}, \mathbf{g} = 0) \quad (15)$$

$$\mathbf{B} = f(\ddot{\mathbf{q}} = \mathbf{q}'', \dot{\mathbf{q}} = \mathbf{q}', \mathbf{q}, \mathbf{g} = 0) \quad (16)$$

$$\mathbf{C} = f(\ddot{\mathbf{q}} = 0, \dot{\mathbf{q}} = 0, \mathbf{q}, \mathbf{g}) \ . \quad (17)$$

Coulomb friction $\mathbf{M}_c = \mathbf{r}_c \ \mathrm{sign}(\mathbf{q}'(\sigma))$, depending on σ, can be added to the term $\mathbf{C}$, while viscous friction $\mathbf{M}_v = \mathbf{r}_v \mathbf{q}'(\sigma)\dot{\sigma}$ corresponds to $\mathbf{D}$. With the abbreviation $z = \dot{\sigma}^2$ (14) leads to a set of $(k = 1 \ldots 6)$ conditions

$$a_k z' + b_k z + c_k + d_k \sqrt{z} = u_k, \quad (18)$$

which can be graphically evaluated in the $[z', z]$ plane and the $[z', z, \sigma]$ space. Notice, when neglecting viscous friction, (18) degenerates to straight lines in the phase plane. However, this simplification can not be done for real systems, since viscous friction is dominant.

B. Formulation of the optimization problem

The global goal of this particular optimization is to find a minimum of the cost function

$$W = \int_0^{t_E} 1 dt, \quad (19)$$

(time optimal solution). However, the unknown cycle time t_E is the sought solution of the optimization, so a transformation of

$$\dot{\sigma} = \frac{d\sigma}{dt} \quad (20)$$

and therefore

$$dt = \frac{1}{\dot{\sigma}} d\sigma = \frac{1}{\sqrt{z}} d\sigma \quad (21)$$

leads to a cost function depending on z of

$$W = \int_0^{\sigma_E} \frac{1}{\sqrt{z}} d\sigma, \quad (22)$$

with well known σ_E. Auxiliary conditions, see (10), for the optimizations can be transformed to

$$-u_{k,max} \leq u_k \leq u_{k,max} \quad (23)$$

$$-\dot{q}_{k,max} \leq q'_k \dot{\sigma} \leq \dot{q}_{k,max} \quad (24)$$

$$-\ddot{q}_{k,max} \leq q''_k z + \frac{1}{2} q'_k z' \leq \ddot{q}_{k,max} \quad (25)$$

$$-v_{max} \leq |\mathbf{z}'_E| \dot{\sigma} \leq v_{max} \quad (26)$$

where in (26), additional constraints for the maximum end-effector velocity are included.

The inequalities (23) represent limiting curves in the $[z', z]$ plane. For each discrete point on the path σ_i there exists a polygon limiting the allowed region, see Fig. 3. Restrictions in the end-effector velocity v_{max} (26) and the joint velocities $\dot{q}_{max}$ (24) can be rewritten to

$$z_{Gr} = \min \begin{bmatrix} \frac{\dot{q}^2_{k,max}}{q'^2_k} \\ \frac{v^2_{max}}{|\mathbf{z}'_E|^2} \end{bmatrix},$$

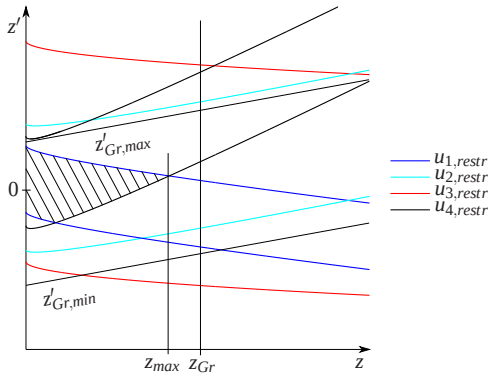


Fig. 3. Restrictions and feasible region

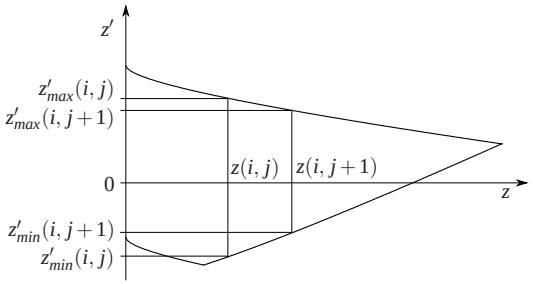


Fig. 4. Expansion of the feasible region

representing a vertical line in the $[z', z]$ - plane. Joint accelerations $\ddot{q}_{max}$ can be taken into account by reformulating (25) to

$$z'_{Gr,min} = \frac{2(-\ddot{q}_{k,max} - q''_k z)}{q'_k} \quad (27)$$

$$z'_{Gr,max} = \frac{2(\ddot{q}_{k,max} - q''_k z)}{q'_k}, \quad (28)$$

leading to additional restrictions in the phase plane.
The maximum value of z in the allowed region, computed at each discrete point σ_i leads to the limiting curve z_{max}, serving as basis for the optimization.

C. Bellman Optimization

The calculation of the optimal trend of the velocity profile z is done with the help of the Bellman optimality principle [4]

> "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."

Therefore the calculation of the optimal trend z_{opt} starts at the last point σ_E by evaluating the maximum and minimum allowed values of z' (z'_{max}, z'_{min}) with the help of the feasible region in Fig. 4. At first we discretize the velocity at each path point σ_i into (m+1) values $z_j(\sigma_i)$ with $j \in (0..m)$. The discretized cost function from (22) equals as a consequence

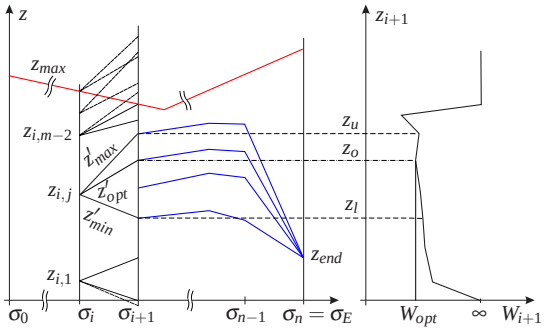


Fig. 5. Bellman optimization

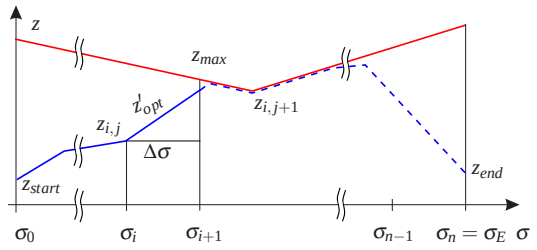


Fig. 6. Computing the solution

of the Bellman optimality principle

$$W_{i,j} = \Delta\sigma \frac{1}{\sqrt{z_{i,j}}} + W_{i+1,j}, \tag{29}$$

where the discretization step size of σ is $\Delta\sigma = \sigma_{i+1} - \sigma_i$. The optimal values of z' can now be evaluated by calculating the highest reachable point $z_u = z_{i,j} + z'_{max}\Delta\sigma$ and the lowest one $z_l = z_{i,j} + z'_{min}\Delta\sigma$, wherein the minimum of the cost function has to be sought, as shown in Fig. 5. This minimum search is done with the method of the golden ratio. With the found location of the minimum z_o, the optimal value of z' can be calculated with

$$z'_{opt}(i,j) = \frac{z_o - z_{i,j}}{\Delta\sigma}. \tag{30}$$

As $z'_{opt}(i,j)$ is calculated for every discretization point (i,j) $i = 0 \ldots n,\ j = 0 \ldots m$, z_{opt} can be evaluated by solving

$$z_{i,opt} = z_{i-1,opt} + z'_{i-1,opt}\Delta\sigma \qquad i = 1..n \tag{31}$$

with a forward iteration, starting at $z_0 = z_{start}$, see Fig. 6 for a graphical interpretation.

As soon as the optimal trend $\dot{\sigma}^2_{opt}$ is evaluated, the time behavior of σ can be calculated by integrating (21)

$$t(\sigma) = \int_0^{\sigma} \frac{1}{\sqrt{z}} d\sigma \tag{32}$$

and calculating the inverse function. Therefore the overall trajectory as well as the feed-forward torques (14) are defined.

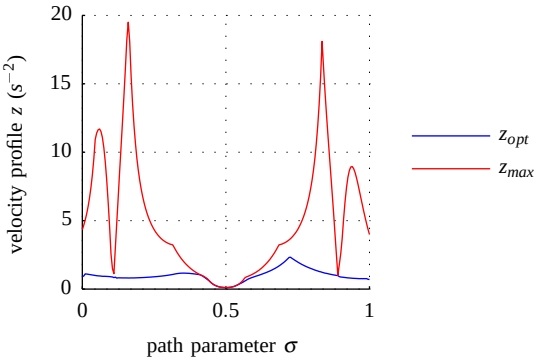


Fig. 7. Limiting curve z_{max} and optimal trend of z_{opt}

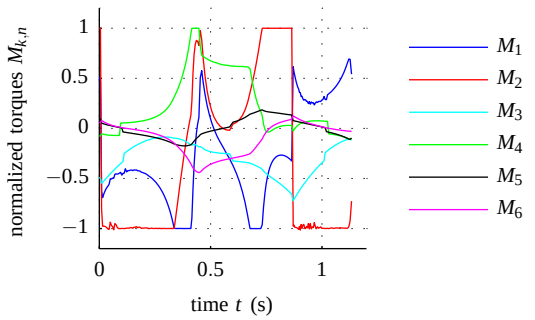


Fig. 8. Normalized torques without acceleration constraints

D. Results

The verification of the proposed optimization strategy is done for a spatial path that interpolates the points

$$\mathbf{P} = \underbrace{\begin{bmatrix} 0.6 & 0.62 & 0.7 & 0.62 & 0.6 \\ 0.8 & 0.75 & 0 & -0.75 & -0.8 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix}}_{\mathbf{p}_0 \quad \mathbf{p}_1 \quad \mathbf{p}_2 \quad \mathbf{p}_3 \quad \mathbf{p}_4}. \tag{33}$$

Fig. (7) presents the limiting curve z_{max} and the optimal trend z_{opt}. The corresponding torques, normalized with respect to the maximum values $M_{k,n} = \frac{M_k}{M_{k,max}}$, are shown in Fig. 8, while the normalized joint angular velocities can be seen in Fig. 9.

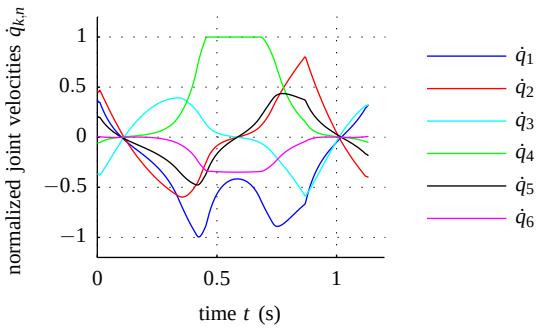


Fig. 9. Normalized joint velocities without acceleration constraints

Comparing Fig. 8 and Fig. 9 one can see, that always one quantity is in the boundary. In the middle of the trajectory the joint angular velocity boundary $\dot{q}_{4,min}$ is active, while in the rest of the time the torque constraints are satisfied. For the given path the robot needs $t_E = 1.13s$.

V. WAITER-MOTION-PROBLEM

The challenge of the "Waiter-Motion-Problem" is to additionally limit the acceleration of an object at the end-effector, that the stiction holds the object on its position. This sticking condition can be written as

$$\sqrt{{}_E a_x^2 + {}_E a_y^2} \leq {}_E a_z \mu_0 \tag{34}$$

with the accelerations of the object

$${}_E\mathbf{a} = \frac{1}{2}\,{}_E\mathbf{J}_\sigma z' + {}_E\mathbf{J}'_\sigma z - \mathbf{A}_{EI}\mathbf{g}, \tag{35}$$

where ${}_E\mathbf{J}_\sigma$ is the Jacobian mapping the velocities $\dot{\sigma}$ to the velocities of the object. $\mathbf{A}_{EI}$ is the rotation matrix from the inertial system to the local coordinate system of the object transforming the gravitational acceleration $\mathbf{g}$ into this coordinate system. The highly nonlinear sticking condition (34) can be treated in the optimization procedure by discretizing the friction cone with $\varphi_i = \frac{i\pi}{2^{p-1}}$ with $i = 1 \ldots 2^{p-1} - 1$. This leads to a set of inequalities

$$|{}_E a_x| \leq {}_E a_z \mu_0 \tag{36}$$

and

$$\left| -\frac{{}_E a_x}{\tan(\varphi_i)} + {}_E a_y \right| \leq \frac{{}_E a_z \mu_0}{\sin \varphi_i} \tag{37}$$

where p denotes the discretization variable. With a higher value of p the exactness of the discretization raises, but also the computation time. Proven values are $p = 5 \ldots 7$. The discretization leads to 2^{p-1} linear upper (subscript u) and exactly as many lower (subscript l) boundary conditions

$$k_l z + d_l \leq z' \leq k_u z + d_u, \tag{38}$$

where the parameters k_l, d_l, k_u, d_u can be calculated by reformulating (36) and (37) in combination with (35). This set of constraints, combined with the physical constraints from (18), yields to a new feasible region in the $[z, z']$ plane and the optimization can be performed to get the time optimal movement for the "Waiter-Motion-Problem".

VI. OPTIMIZATION OF THE ORIENTATION

Obviously, a solution for the "Waiter-Motion-Problem" with constant end-effector orientation is not time optimal for the movement of an object. It can be improved by additionally optimizing the orientation. For this case, rotations about the local ${}_E x$ axis by an angle α and rotations about the local ${}_E y$ axis by an angle β are incorporated into the optimization procedure, see Fig.10 for the definition of the rotations. The angles α and β are again parametrized by splines and discretized with σ. Starting with a constant value of zero, the optimizer varies the spline interpolation points. Since the rotations have to be limited, a nonlinear solver that is able to deal with constraints is necessary. For the solution, the *fmincon* function from the Optimization Toolbox within Matlab is utilized.

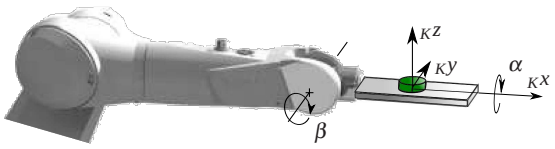


Fig. 10. Definition of the angles α and β

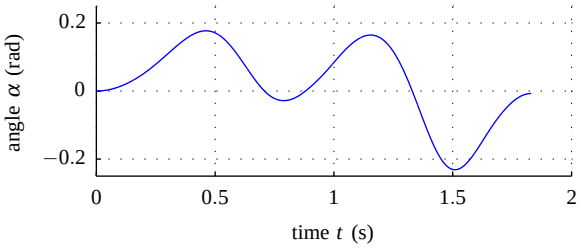


Fig. 11. Variation of α over the path parameter σ

VII. RESULTS

The optimization is evaluated again for the path defined in (33) extended by the orientation variation. Fig. 11 exemplary shows the time behavior of the angle α. Since a path is chosen, that passes near the wrist singularity ($q_5 = 0$) of the robot at time $0.7s$, the end-effector velocity decreases drastically. This can be compensated with α. The following acceleration and deceleration phase of the end-effector lead to a variation of the angle α. The corresponding joint velocities can be seen in Fig. 12 and Fig. 13. The wrist singularity is indicated by the bounded velocity $\dot{q}_4$.
From now on, the focus should be on a comparison between the solutions using constant and optimized orientation. For the trajectory with constant orientation $2.19s$ are needed, while the trajectory with optimized orientation only requires $1.82s$. The difference in time between the optimized and constant orientation is $0.37s$ which is about 17%. The time optimal solution of the robot without any object on the tray, as described in section IV, lasts about $t_E = 1.13s$. So most of the time the boundaries for the sticking condition are active, see Fig. 14 and Fig. 15 for the normalized acceleration of the object. Due to the multiple discretization of the friction

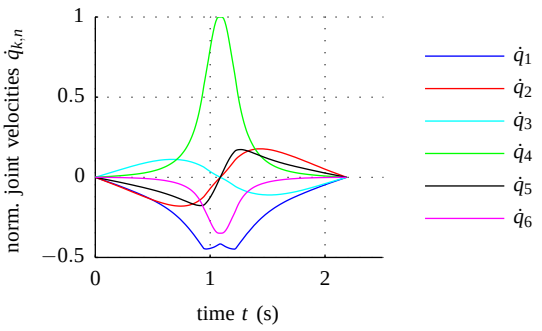


Fig. 12. Normalized joint velocities for constant orientation

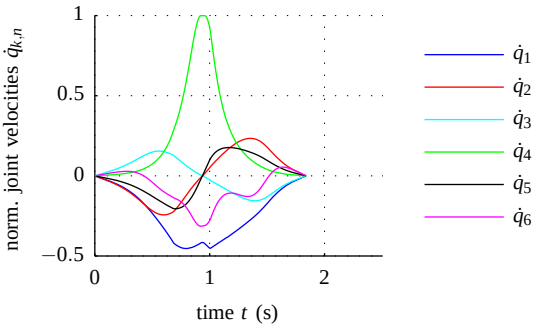


Fig. 13. Normalized joint velocities for variable orientation

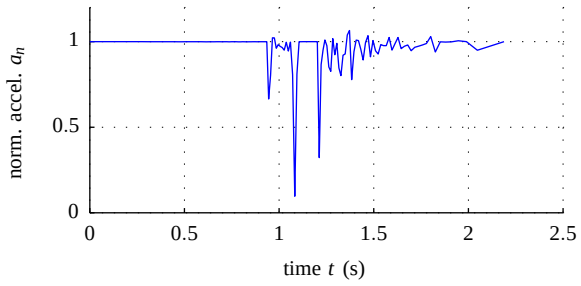


Fig. 14. Normalized acceleration on the object for constant orientation

cone (38) the acceleration constraint is exceeded in some phases, as shown in Fig 15. To counteract this discretization influence, the value of the dry friction coefficient can be adapted. A solution for this adaptation delivers Fig. 16, where μ_0 indicates the gradient of the straight line. Adapting the dry friction coefficient with $\Delta\mu$ results in a change of the gradient. To ensure that no slippage of the object occurs, μ_0 is changed to a value so that the entire curve of the occurring accelerations lies in the left upper plane. Accepting a higher calculation time, the value of $\Delta\mu_0$ decreases with increasing discretization p.

For the practical implementation on the Stäubli Rx130, a value of $p = 6$ yields acceptable results. For demonstration purposes a comparison of the optimal trajectory with optimized orientation, without optimized orientation but with the same cycle time and an optimal trajectory without optimized orientation is done. As result one can see, the object starts

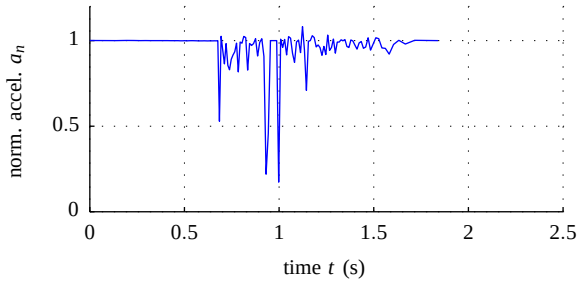


Fig. 15. Normalized acceleration on the object for variable orientation

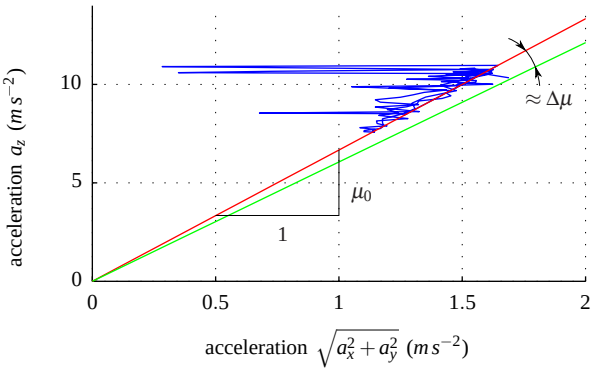


Fig. 16. Adaption of the dry friction coefficient μ_0

sliding in the second case. In cases one and three the object sticks on the tray, but needs noticeable more time to overcome the path in case three.

VIII. CONCLUSION

This work is about time optimal path planning for industrial robots under consideration of various constraints. As case example the ”Waiter-Motion-Problem” is treated. To be able to handle this problem a spatial path has to be defined. A spline parametrization for a set of points in world coordinates is used. With a transformation of the equations of motion of the robot into the phase plane, a time optimal solution is evaluated using the Bellman optimal principle. The advantage of this method is, that arbitrary cost functionals can be optimized. So we are not restricted to the time optimal case. Time/energy optimal solutions are also of great interest. The method is enhanced by additionally optimizing the orientation of the end-effector. This leads to a significant reduction of the overall motion time.

ACKNOWLEDGMENT

Support of the present work in the framework of the peer-reviewed Austrian Center of Competence in Mechatronics (ACCM) is gratefully acknowledged.

REFERENCES

[1] F. Geu Flores, A. Kecskeméthy, Time-optimal path planning along specified trajectories. In: H. Gattringer, J. Gerstmayr (eds.): Multibody System Dynamics, Robotics and Control, 2012, pp. 1-15.
[2] C. De Boor, A practical guide to splines, New York: Springer-Verlag 1978.
[3] L. A. Piegl, W. Tiller, The NURBS Book, 2nd ed. Springer-Verlag 1995.
[4] R. E. Bellman, S. E. Dreyfus, Applied Dynamic Programming, Princton Univ. Press, 1962.
[5] F. Pfeiffer, R. Johanni, A concept for manipulator trajectory planning, in IEEE J. Rob. Aut 3 (1987), Nr. 2, pp. 115-123.
[6] H. Bremer, Elastic Multibody Dynamics: A Direct Ritz Approach. Linz, Austria, Springer-Verlag GmbH, 2008, pp 445.
[7] K. Shin, N. McKay, Minimum-time control of robotic manipulators with geometric path constraints, in IEEE Trans. Autom. Control 30 (1985), Nr.6, pp. 531-541.
[8] J. E. Bobrow, S. Dubowsky, J. S. Gibson, Time-optimal control of robotic manipulators along specified paths, in Int J. Rob. Res. 4(1985), pp. 3-17.

Optimal Path-Planning in the Special Case of Ball Throwing Using an Industrial Robot

Thomas Raukamp, Klemens Springer and Hubert Gattringer

***Abstract*— This paper presents optimal path planning in the special case of ball throwing. The task has to be fullfilled by a standard industrial robot with six degrees of freedom, wherefore a parameter identification is conducted. For the purpose of obtaining optimal trajectories fulfilling this exercise, cubic basis splines are used. A nonlinear optimization considering physical constraints as well as the launch angle generates the optimal motion. For the calculation of the balls flight path drag force is included. Results in simulation and at the real robot are shown.**

I. INTRODUCTION

Nowadays robots are used in industry in almost all fields of work such as packaging, assembly, welding, measuring, cutting, etc. An exact path planning is essential in order to perform a certain task perfectly. In this work, the robot is used for an unusual task which it has to fulfill. A ball which is located in the gripper of the robot has to be thrown to a certain destination, which is outside the robots working range.

There are already several studies that have dealt with this topic. Some of these use robots with few degrees of freedom only, see [1] for example. An extended motion generation for throwing a ball with a robot with six degrees of freedom is shown in [2]. In [3] the given task is considered from a completely different perspective. Two variable impedance actuators are used in order to drastically extend the throwing range of the robot approaching human efficiency. Usually the studies are conducted using self-built robots and cubic splines for path planning. In [4] an industrial robot is used. There the flight path is approximated with a parabola. A trapezoid acceleration profile is used for planning a trajectory for the robot to a certain release point of the flight path.

In comparison, here a standard industrial robot is used in combination with cubic basis splines for path planning. To reach a desired position with the selected object, it is necessary that it possesses a certain speed and direction when the gripper opens. In order to minimize the error between the desired and the reached position an optimization is performed. Therefore physical constraints as well as the possible restriction of the launch angle are considered. For improving the hit rate, the robots unknown parameters are identified and air resistance is included in the computation of the flight path.

Johannes Kepler University Linz
Altenbergerstr. 69, 4040 Linz, Austria
thomas@raukamp.at, klemens.springer@jku.at, hubert.gattringer@jku.at
http://www.robotik.jku.at

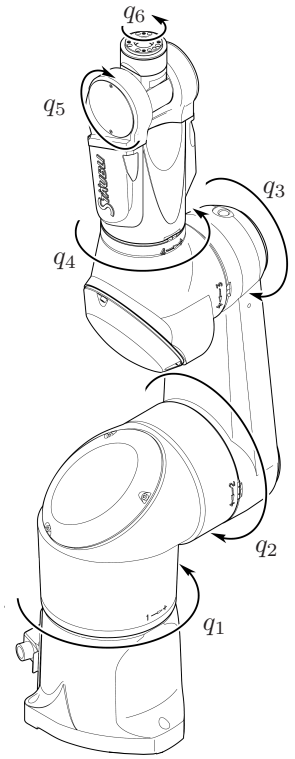


Fig. 1. TX90L - Stäubli

In Section II the used industrial robot, the modeling process and parameter identification are presented. The path planning for the robot trajectory is described in III. Section IV contains the balls flight trajectory calculation. The nonlinear optimization is discussed in V. It addresses the cost function, the constraints and the optimization variables. Section VI shows results of simulation and evaluation on the real robot.

II. MANIPULATOR

In the present work, an industrial robot (TX90L) of the company Stäubli is used (see Fig. 1). This robot consists of six axes described by the coordinates $\mathbf{q} = [q_1, q_2, q_3, q_4, q_5, q_6]^T$. An overview of the physical limits of the TX90L can be seen in Table I. At the sixth axis, a pneumatically actuated gripper is mounted. It is responsible to transport the ball to the throwing position. To determine the position $\mathbf{r}_E(\mathbf{q})$ and velocity $\dot{\mathbf{r}}_E(\mathbf{q})$ of the end-effector a kinematic model is necessary. Additionally a dynamical

TABLE I

TX90L - PHYSICAL LIMITS

Axis	$q_{min}(°)$	$q_{max}(°)$	$\dot{q}_{max}$ (°/s)	M_{max} (Nm)
1	-180	180	400	42
2	-130	147.5	390	42
3	-145	145	400	17.5
4	-270	270	540	4.5
5	-115	140	475	3.4
6	-270	270	760	2.2

model of the robot is calculated to be able to compute the drive torques in dependence of the joint angles and its derivatives, see Subsection II-A. Thus the maximum torques can be taken into account in optimization.
In the dynamical model a few unknown parameters such as the mass of the arms and inertia of the motors strongly influence the resulting accuracy. Therefore the identification of these parameters is discussed in Subsection II-B.

A. Modeling

The robot is considered as a rigid system with $N = 12$ bodies, as each of the six axes consists of one arm and one drive. The equations of motion are calculated according to [5] with the Projection Equation

$$\sum_{i=1}^{N} \mathbf{F}_i^T \left\{ \mathbf{M} \begin{pmatrix} \dot{\mathbf{v}}_c \\ \dot{\boldsymbol{\omega}}_c \end{pmatrix} + \mathbf{G} \begin{pmatrix} \mathbf{v}_c \\ \boldsymbol{\omega}_c \end{pmatrix} - \begin{pmatrix} \mathbf{f}^{es} \\ \mathbf{M}^{es} \end{pmatrix} \right\}_i = 0. \tag{1}$$

The variables occurring here are given for each body i with the functional matrix

$$\mathbf{F}_i^T = \left[\left(\frac{\partial \mathbf{v}_c}{\partial \dot{\mathbf{q}}} \right)^T \left(\frac{\partial \boldsymbol{\omega}_c}{\partial \dot{\mathbf{q}}} \right)^T \right]_i^T,$$

the mass matrix

$$\mathbf{M}_i = \begin{bmatrix} m\mathbf{E} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}^c \end{bmatrix}_i,$$

and the gyroscopic matrix

$$\mathbf{G}_i = \begin{bmatrix} \tilde{\boldsymbol{\omega}}_c m & \mathbf{0} \\ \mathbf{0} & \tilde{\boldsymbol{\omega}}_c \mathbf{J}^c \end{bmatrix}_i,$$

when using body-fixed coordinate systems. The variables $\mathbf{v}_{c_i}$ and $\boldsymbol{\omega}_{c_i}$ in (1) describe the speed and angular velocity of each body. Furthermore the impressed forces $\mathbf{f}_i^{es}$ and torques $\mathbf{M}_i^{es}$ are introduced. Finally the equations of motion can be written in minimal description

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{g}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{Q}$$

with $\mathbf{Q}$ as vector of generalized forces, the mass matrix $\mathbf{M}$ and the nonlinear terms $\mathbf{g}$.

B. Parameter identification

As mentioned before, an exact model is of great importance for this task. For parameter identification, see e.g. [6], the equations of motion are now expressed in terms of its unknown base parameters $\mathbf{p}_B$:

$$\boldsymbol{\Theta}\mathbf{p}_B = \mathbf{Q}. \tag{2}$$

This results in the matrix of information $\boldsymbol{\Theta}$, the vector of unknown base parameters $\mathbf{p}_B$ and the generalized forces $\mathbf{Q}$. The method of least squares is used to determine the unknown parameters

$$\mathbf{p}_B = \left[\boldsymbol{\Theta}^T \boldsymbol{\Theta} \right]^{-1} \boldsymbol{\Theta}^T \mathbf{Q}. \tag{3}$$

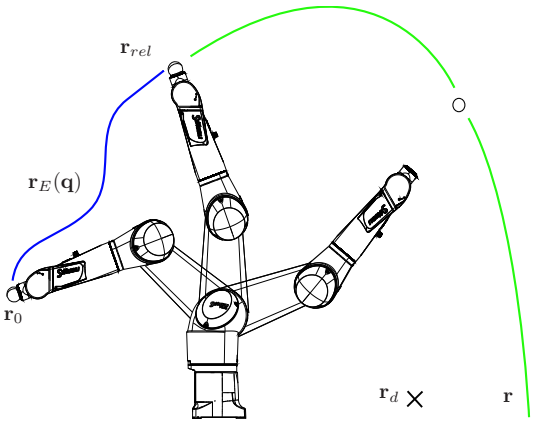


Fig. 2. Overall view

III. PATH PLANNING

The task of the robot is to move from a given starting point $\mathbf{r}_0$ to the point $\mathbf{r}_{rel}$ where it releases the ball, see Fig. 2. To plan this path basis splines according to [7] are used. These have the advantage that the modification of a control point, the so-called de Boor point, only results in a local change. With these splines paths are planned for all degrees of freedom $\mathbf{q} = [q_1, q_2, q_3, q_4, q_5, q_6]^T$ of the robot.

A. Cubic B-spline curve

Before a path can be planned with basis splines, it is necessary to determine the so-called basis functions. These are calculated using the recursion formula

$$N_j^d(t) = \frac{t - t_j}{t_{j+d} - t_j} N_j^{d-1}(t) + \frac{t_{j+d+1} - t}{t_{j+d+1} - t_{j+1}} N_{j+1}^{d-1}(t) \tag{4}$$

and

$$N_j^0(t) = \begin{cases} 1 & t \in [t_j, t_{j+1}[\\ 0 & otherwise \end{cases}.$$

As cubic basis splines are used with the degree $n = 3$, the expression $N_j^3(t)$ is necessary. There are m real valued knots called t_i, with

$$t_0 \leq t_1 \leq ... \leq t_{m-1}.$$

The basis functions span a range $[t_j, t_{j+4}]$ of $n + 2$ knots. With the specification of control points, the path can be

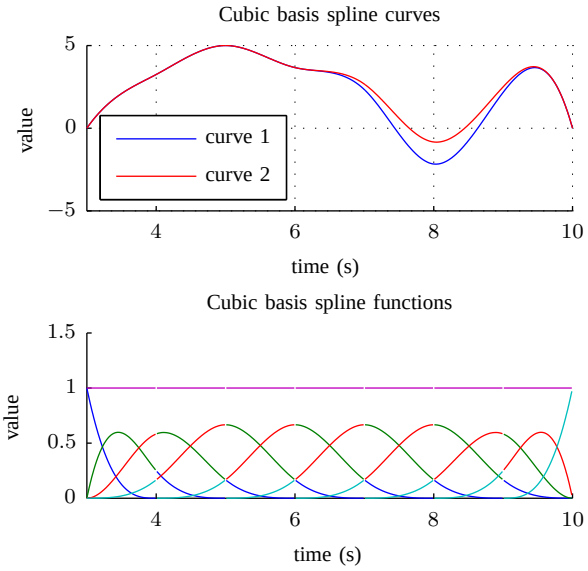


Fig. 3. Cubic basis spline

calculated. Each control point is assigned to a basis function, thus the path can be determined by the formula

$$s(t) = \sum_{j=0}^{m-n-2} d_j N_j^3(t). \tag{5}$$

In the upper section of Fig. 3 two cubic curves which differ only at one control point are shown. Obviously, a local change appears only in this area. The rest of the curve remains unchanged. In the lower part of Fig. 3 the cubic basis function and the sum of these are shown. It can be seen that the sum of the basis functions has to be one what is described as a local decomposition of one.

B. Orientation

At the release point $\mathbf{r}_{rel}$ the ball should move unaffected by the gripper with its impressed speed to its desired position $\mathbf{r}_d$. In order to reach this goal the final orientation of the end-effector has to be adjusted. Thus the angles $[q_4, q_5, q_6]$ are calculated in dependence of q_1, q_2 and q_3. The resulting orientation matrix of this angles is generally

$$A_{IE} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

and held constant during the whole trajectory at this desired release orientation. The required angles are obtained with

$$\begin{aligned} q_6 &= atan2\left(\frac{a_{11}}{a_{21}}\right) \\ q_5 &= atan2\left(\frac{sin(q_6)a_{11} + cos(q_6)a_{21}}{a_{31}}\right) \\ q_4 &= atan2\left(\frac{-cos(q_6)a_{12} + sin(q_6)a_{22}}{cos(q_6)a_{13} - sin(q_6)a_{23}}\right) \end{aligned} \tag{6}$$

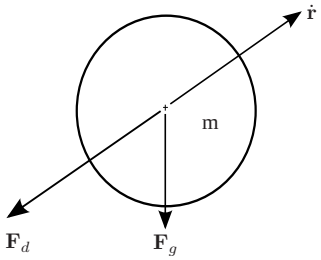


Fig. 4. Model of the ball

and are hence not optimized so far. The lost optimization benefit is estimated as significantly small but results in a faster calculation.

IV. Ball flight path

In order to calculate the flight path of the ball from the release point $\mathbf{r}_{rel}$ to $\mathbf{r}$, it is necessary to specify an appropriate model for the ball. This is based on illustration in Fig. 4. In that case the weight force

$$\mathbf{F}_g = m \begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix}^T,$$

the drag force $\mathbf{F}_d$ and the momentum of the ball are considered. The principle of linear momentum leads to the expression

$$m\ddot{\mathbf{r}} = \mathbf{F}_d + \mathbf{F}_g. \tag{7}$$

The drag force

$$\mathbf{F}_d = -\frac{1}{2} c_w \rho A \dot{\mathbf{r}} \left\|\dot{\mathbf{r}}\right\|$$

contains the flow resistance coefficient c_w, the density of air ρ, the velocity $\dot{\mathbf{r}}$ of the ball and the surface A. This formula is converted to $\mathbf{F}_d = -m\alpha\dot{\mathbf{r}} \left\|\dot{\mathbf{r}}\right\|$ with the mass m of the body. α is describing the effect of air resistance. The result is a mathematical model in state space

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{r}_x \\ \dot{r}_y \\ \dot{r}_z \\ -\alpha \left\|\dot{r}\right\| \dot{r}_x \\ -\alpha \left\|\dot{r}\right\| \dot{r}_y \\ -\alpha \left\|\dot{r}\right\| \dot{r}_z - g \end{pmatrix} \tag{8}$$

with the state vector $\mathbf{x} = [r_x\, r_y\, r_z\, \dot{r}_x\, \dot{r}_y\, \dot{r}_z]^T$, the coordinates $(r_x\, r_y\, r_z)$ and the velocities $(\dot{r}_x\, \dot{r}_y\, \dot{r}_z)$ of the ball. Utilizing (8) and a Runge-Kutta time integrator, the balls flight path is calculated.

V. Optimization

This section discusses the optimization that computes the path for the robot. Initially, a desired target point $\mathbf{r}_d$ is selected. This is achieved by entering the world coordinates

(x, y, z), whereby the robot is in the center of the coordinate system. The robots working range is divided into four quadrants. Using this a suboptimal starting point $\mathbf{r}_0$ is selected in the opposite quadrant of $\mathbf{r}_d$. An initial trajectory completely defind by de Boor points is given for the optimization. Except for the starting point of the optimization these de Boor points are varied. The optimization affects the flight phase with the position of the ball $\mathbf{r}$ that may approximate the desired target $\mathbf{r}_d$. The distance between the positions $\mathbf{r}$ and $\mathbf{r}_d$ should be minimized. Simultaneously certain physical limitations and other restrictions are considered.

A. Optimization variables

The optimization of the robot trajectory is done in configuration space and now described in more detail. There will be one path planned for each of the six joint angles. Each path consists of a starting point and N additional control points. These are optimized for the axes q_1, q_2 and q_3. The trajectories for the other axes q_4, q_5 and q_6 are resulting from the calculated orientation, see Subsection III-B. In this case, there are $m = 3N$ optimization variables combined in the vector $\mathbf{c} = [d_1 ... d_{3N}]$. The starting point $\mathbf{r}_0$ and the control points in $\mathbf{c}$ describe the paths for the joints q_1, q_2 and q_3.

B. Cost function

In order that an optimization can fulfill this task, a objective function is set up. The following objective function

$$\min_{\mathbf{c}\in\mathbb{R}^m} \frac{1}{2}c_1 \left(\mathbf{r}_d - \mathbf{r}(\mathbf{c})\right)^2 + \frac{1}{2}c_2 \int \left(\mathbf{Q}^T(t,\mathbf{c})\mathbf{Q}(t,\mathbf{c})\right)^2 dt \quad (9)$$

is used. The distance between the desired target $\mathbf{r}_d$ and the nearest throwing position $\mathbf{r}(\mathbf{c})$ shall be minimal and require as little energy as possible. The position $\mathbf{r}(\mathbf{c})$ is obtained from the flight path of the ball, whereby the shortest distance to the target

$$|\mathbf{r}_d - \mathbf{r}(\mathbf{c})| = \min_{i=1...n} |\mathbf{r}_d - \mathbf{r}_i(\mathbf{c})|$$

is used for the cost function in (9). Therefore the trajectory of the ball is discretized in n points. Because the criteria may be weighted differently, the variables c_1 and c_2 are demanded.

C. Constraints

It is important to note that different physical limitations must be considered:

$$\begin{array}{rcll} q_i(t,\mathbf{c}) & \leq & q_{max,\,i} & i = 1..6 \\ q_i(t,\mathbf{c}) & \geq & q_{min,\,i} & i = 1..6 \\ |\dot{q}_i(t,\mathbf{c})| & \leq & \dot{q}_{max,\,i} & i = 1..6 \\ |Q_i(t,\mathbf{c})| & \leq & Q_{max,\,i} & i = 1..6 \end{array} \quad (10)$$

These are the limited rotations of the axes, the maximum joint speeds and the maximum drive torques. As the ball may be catched at the desired position, a flat and therefore fast trajectory should be avoided. An additional condition for the launch angle

$$\phi = atan\left(\frac{\dot{r}_z}{\dot{r}_{xy}}\right) \quad (11)$$

(see Fig. 5) is introduced. This launch angle is limited to

$$25^\circ \leq \phi \leq 65^\circ.$$

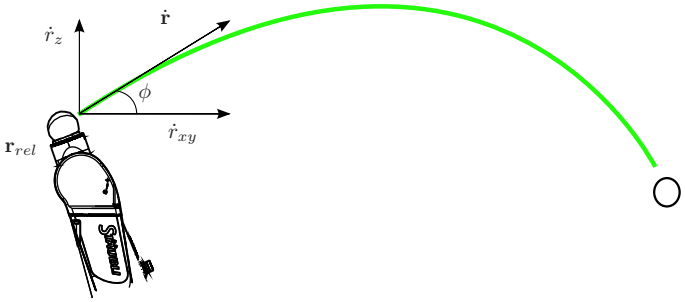


Fig. 5. Launch angle

D. Deceleration path

After the robot reaches the release position $\mathbf{r}_{rel}$ the flight path of the ball begins. The decelerating trajectory is also calculated with an optimization. The angle, the angular velocity and the angular acceleration have to be continuous. By specifying three control points $[d_0,\, d_1,\, d_2]$ this can be achieved. These points are determined by

$$\begin{array}{rcl} d_0 & = & q(t_{end}) \\ d_1 & = & \frac{\Delta t_1}{3}\left(\dot{q}(t_{end}) + \frac{3}{\Delta t_1}d_0\right) \\ d_2 & = & \frac{(\Delta t_2 + \Delta t_1)(\Delta t_1)}{6}\left(\ddot{q}(t_{end}) - \frac{6d_0}{(\Delta t_1)^2} + \right. \\ & & \left. \left(\frac{6}{(\Delta t_1)^2} + \frac{6}{(\Delta t_2 + \Delta t_1)(\Delta t_1)}\right) d_1\right) \end{array} \quad (12)$$

with the difference of the knots $\Delta t_1 = t_1 - t_0$ and $\Delta t_2 = t_2 - t_1$. The path consists of these three points, and N further points. Now, a path is planned for all six axes in joint space. Thus there are $m = 6N$ optimization variables. The used cost function and the restrictions are

$$\begin{array}{rcl} \multicolumn{3}{c}{\min_{\mathbf{c}\in\mathbb{R}^m} \frac{1}{2}c_3 \int \left(\dot{\mathbf{q}}(t,\mathbf{c})^T \dot{\mathbf{q}}(t,\mathbf{c})\right) dt} \\ s.t. \quad q_i(t,\mathbf{c}) & \leq & q_{max,\,i} \quad i = 1..6 \\ q_i(t,\mathbf{c}) & \geq & q_{min,\,i} \quad i = 1..6 \\ |\dot{q}_i(t,\mathbf{c})| & \leq & \dot{q}_{max,\,i} \quad i = 1..6 \\ |Q_i(t,\mathbf{c})| & \leq & Q_{max,\,i} \quad i = 1..6. \end{array} \quad (13)$$

VI. Results

The optimization was performed for several targets, achieving the same accuracy for each case. Exemplarily the results

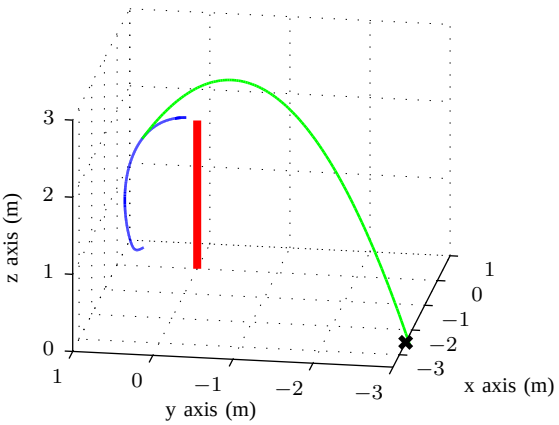


Fig. 6. Optimization result

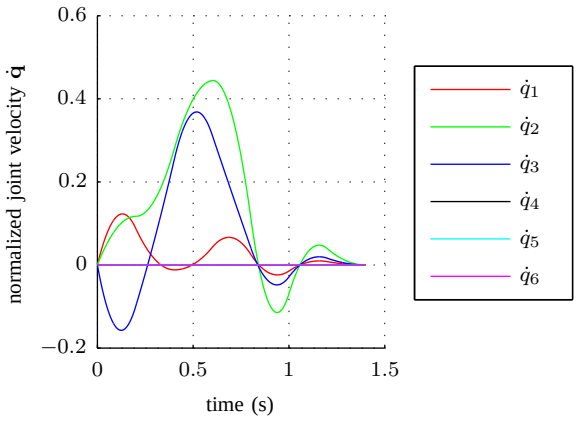


Fig. 8. Normalized joint velocity

of the optimization and performance for one target are shown in this Section.

A. Simulation

The desired target position $\mathbf{r}_d$ is given in meters with the coordinates (-2.5, -3, 0). Four control points have been specified for the acceleration as well as for the deceleration path, thus there are 24 optimization variables. The result of the optimization is shown in Fig. 6. The red bar shows the robot in a stretched position. The blue curve denotes the calculated robot path. The green curve shows the trajectory of the ball. A black cross marks the desired position $\mathbf{r}_d$. The initial trajectory results in a shortest distance to the target of 3,023 m. With the optimization a distance of only 0.12 mm is obtained after 30 iterations, at a launch angle ϕ of 26.6°. The required angle curves and angular velocities are shown in Fig. 7 and 8.

The necessary drive torques can be seen in Fig. 9. Both the angular velocities as well as the torques are presented normalized with respect to the maximum values which can be seen in Table I.

B. Performance

The optimization is done in MatLab using a SQP optimization routine. The resulting trajectory is implemented using Lookup-Tables. The robot is controlled by a B&R industrial PC running with a cycle time of 400 μs. In Fig. 10 the throwing sequence is shown. Picture 1 shows the robot in its starting position with the gripped ball.

In 2 and 3, the robot moves to the release position where the gripper opens and the ball is thrown. In 4, the robot has arrived at its final position. In 5-8 the ball moves to the desired target. The error between the desired and the reached position is 6 cm in x-direction and 7 cm in y-direction. This results in a deviation of approximately 2.4 % and 2.3 %.

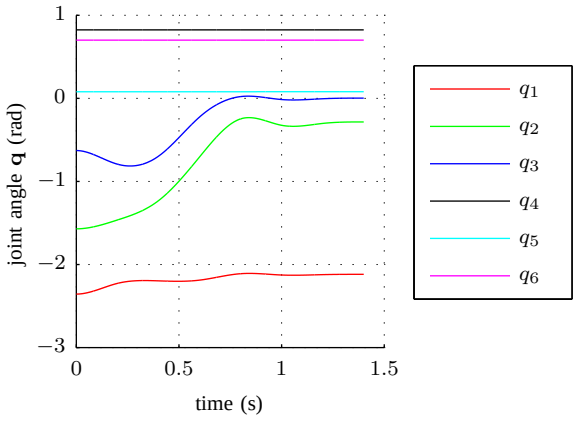


Fig. 7. Joint angle

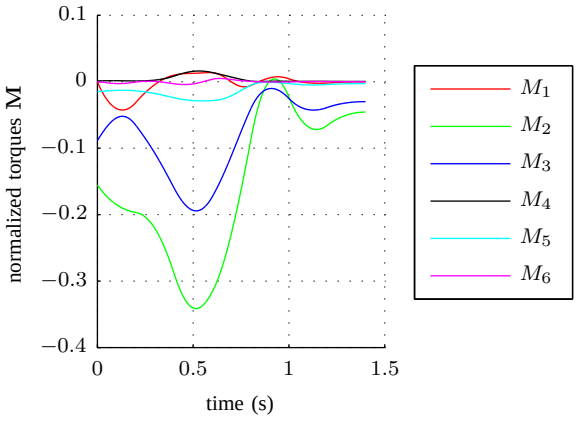


Fig. 9. Normalized torques

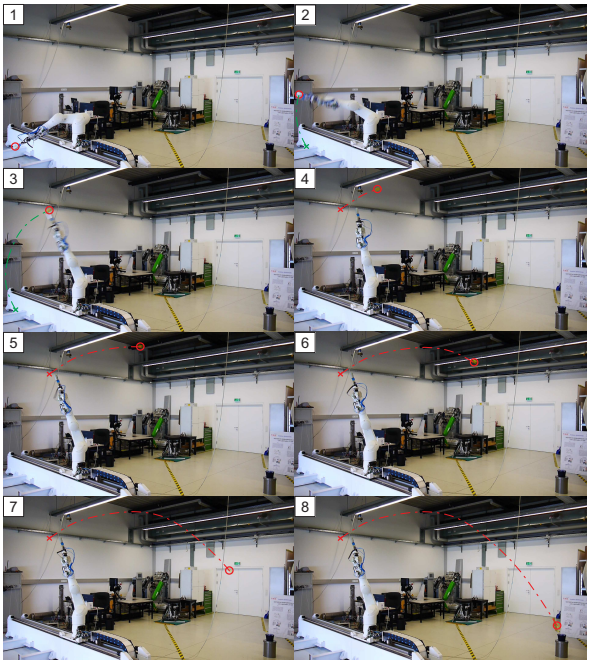


Fig. 10. Throw sequence

VII. CONCLUSIONS AND FUTURE WORK

In this contribution path planning for a standard six-axes industrial robot has been developed using the method of cubic basis splines. These paths are optimized to throw a ball to a certain position best possible. Additionally a parameter identification was done to take the robots physical constraints into account.

In future work an existing camera system can be used to determine the desired target position. With two cameras already mounted on the ceiling, a position can be determined and passed to the optimization. Furthermore the trajectory of the ball can be tracked. This should lead to improvements in the ball model. Additionally a combination of ball throwing and ball catching, implemented in [8], is imaginable.

ACKNOWLEDGMENT

Support of the current work within the framework of the Austrian Center of Competence in Mechatronics (ACCM) is gratefully acknowledged.

REFERENCES

[1] N. Kato, K. Matsuda, and T. Nakamura. Adaptive control for a throwing motion of a 2 DOF robot. In Proc. of the 4th International Workshop on Advanced Motion Control, AMC 96-MIE, volume 1, 1996

[2] F. Lombai: Throwing motion generation using nonlinear optimization on a 6-degree-of-freedom robot manipulator. In: Proceeding IEEE International Conference on Mechatronics, 2009

[3] D. J. Braun et al.: Optimal Torque and Stiffness Control in Compliantly Actuated Robots. In: Proceeding IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012

[4] W. August, S. Waeldele, B. Hein and H. Woern: Accurate object throwing by an industrial robot manipulator. In: Proceeding Australasian Conference on Robotics & Automation, 2010

[5] H. Bremer: Elastic Multibody Dynamics: A Direct Ritz Approach. Linz, Austria: Springer-Verlag GmbH, 2008

[6] W. Khalil and E. Dombre: Modeling, Identification and Control of Robots. London: Kogan Page Science, 2004

[7] C. de Boor: A practical guide to splines. New York: Springer, 1978

[8] M. Neubauer: A Ball-Catching Redundant Industrial Robot. The 13th Mechatronics Forum International Conference, Vol. 2, Trauner Verlag, Linz, pp. 462-468, 2012

RoboCup Junior Soccer Demo League

Georg Richter, Madeleine Redl, Dawn Alolino, Sandra Dertnig, Alexander Hofmann,
University of Applied Science Technikum Wien

***Abstract*— The aim of our project is the development of a new soccer league for RoboCup Junior. The RoboCup Junior soccer arena is equipped with two cameras, which are connected to a computer beside the arena. For processing the data of the cameras, we use the Vision system of the Small Size League. The teams are able to send control commands to the robots by using a Bluetooth connection.**

I. Introduction

We are second year computer science students at UAS Technikum Wien. In this paper we highlight the current state of an on-going project of designing a new soccer league for RoboCup Junior which aims to solve some of the problems encountered in the existing leagues.

II. On RoboCup Junior

The RoboCup Organization aims to promote robotics and AI programming. The ultimate goal of the RoboCup is to create a human-like soccer team that can beat the human world champions. In order to reach that goal regional and international competitions are held to get people invested in robotics and programming. [7]

The RoboCup Junior as a part of the RoboCup Organisation is created for young students up to the age of 19. The aim of the RoboCup Junior is to provide children an opportunity to getting to know the field of robotics. There are currently three RoboCup Junior leagues: Dance, Rescue and Soccer [1].

III. Motivation

In Austria most junior teams mostly use the Lego Mindstorms platform because it is relatively cheap and easy to program. However one of the major flaws of this platform is insufficient sensory input and processing. Most advanced sensors have to be bought extra and advanced features such as image processing are simply not available. And while it is one of the goals of the RoboCup to create new sensors as well, most teams in RoboCup Junior will not have the opportunity or ability to do so.

To counter this we have decided to adapt one of the existing RoboCup Soccer leagues – the Small Size League – for use in the RoboCup Junior. While robots in the junior league have to rely solely on their own sensors and processors, in Small Size League there are cameras above the field which track the position of the Robots as well as the ball and sends it to the team's computer, where a program coordinates the robots accordingly. This allows the teams to focus much more on effective tactics rather than sensor processing while keeping a low price point and easy setup.

IV. Settings

A. Teams

A team consists of two robots. The team color will be assigned before the game starts. It can be either blue or yellow. The team color is equal to the center marker color of the team pattern.

B. Arena

The playing field has the same dimensions as the field for the existing junior soccer league which is 183 by 122cm with an outer area of 30 cm width [3]. Two cameras are mounted 1.6 meters above the field. One camera captures the image of one field half. The images are then sent to a Computer which runs the Vision Software for Small Size League [4].

C. Patterns

In order to detect and identify the robots in the arena, each robot must have a unique pattern. To keep the designing of the patterns simple, the teams will use the Small Size League Vision standard pattern. Moreover, there will be also a given set of possible patterns.

D. Connection

Just like the Small Size League the new league is going to use the referee box [5] for keeping records of the timeouts, time remaining, score and penalties. For this reason we will offer the teams a Java library to receive the commands. For the connection to the robots, the teams will have to use Bluetooth. Since we want the teams to concentrate on programming the AI, we will provide a class to send commands to the robots and another one to receive those commands.

V. The course of a game

Before a game can start each team will be assigned a team color, which is either blue or yellow. In order to distinguish the teams, the blue team has to use a pattern with a blue center marker and the yellow team must have a pattern with a yellow center marker. Like in the SSL the robots must fit within a 180mm diameter circle and are not allowed to be higher than 150mm.

The game will start when the robots are placed within the soccer arena and the team members are ready to control their robots with an off-field computer.

A. Robots and ball detection

All objects within the arena are tracked by two cameras, which are installed in a height of 1.6 meters above the field.

Each camera tracks the objects of one field half and is connected to a computer. The Small Size League Vision software on the computer processes the camera information and detects and identifies the objects on the field. The program sends the detection results via UDP multicast. The computer and the configured Small Size League Vision software will be provided for the teams.

B. *Receiving detection results*

In order to receive the detection information from the Vision system, it is the task of the teams to create a client application, which is capable for receiving the UDP packets from the Vision. Since the packets are encoded using Google Protocol Buffers [6], the packets have to be parsed before the information can be read. We will provide the classes for parsing the packets.

After parsing the received packets, the teams can read the detection data. The data includes among others the positions and the directions of the robots and position of the ball.

C. *Referee Box*

…is a PC program designed to provide a GUI to the assistant referee, which sends commands to the competing teams. The referee box will keep a record of the timeouts, time remaining, score and penalties. [4]

The teams receive the commands of the referee box continuously via UDP multicast datagrams.

D. *Sending commands to robots*

We have created a simple Java class to send commands from the PC to the robots as well as a class for the robots to receive and process the data coming from the PC. They will be available for download. However, we do not want to force the teams to use them because using our Java classes requires leJOS NXJ.

VI. Future Work

Our immediate goal for the next few months is to create two exemplary programs to demonstrate the viability as RoboCup Junior League. Furthermore we are currently developing several simple and easy to use interfaces between the Vision software, the Robots and controlling computer which would be publicly available. These interfaces will be based on C++, because it is much more commonly known by potential RoboCup Junior candidates. This will make it easier for younger students with little knowledge of Bluetooth or the underlying Java software to compete in our new league.

References

[1] RoboCup Junior Official Web Site. http://rcj.robocup.org/

[2] RoboCup Small Size League Official Web Site http://robocupssl.cpe.ku.ac.th

[3] RoboCup Junior Soccer Rules as of 2013 http://rcj.robocup.org/rcj2013/soccer_2013.pdf For field size see '3.2 Dimensions of the field' and 'Field diagram'

[4] RoboCup Small Size League Shared Vision System. http://robocupssl.cpe.ku.ac.th/sslvision, https://code.google.com/p/ssl-vision/

[5] RoboCup Small Size League Referee Box. http://robocupssl.cpe.ku.ac.th/referee:start

[6] Google Protocol Buffers https://developers.google.com/protocol-buffers/?hl=de-DE

[7] Official RoboCup Web Site http://www.robocup.org/

Flexible Assistance System for Packaging Electronic Consumer Goods using Industrial Robots*

Martijn Rooker, Alfred Angerer, Frank Wallhoff, Jürgen Blume, Alexander Bannatt, Paolo Ferrara, Aitor Olarra, Janne Kiirikki, Andreas Pichler

Abstract— **Automated packaging is becoming more and more interesting for production sites. Unfortunately, in many industrial companies, the packaging process is still performed by a human workforce, resulting in workplaces that are dangerous, tedious or non-ergonomic. Many solutions have been introduced to improve the packaging process, but most of them most of these processes only include small and light goods. In this paper, solutions from a research project are presented that support the human worker while packaging mid- to uppersized electronic consumer goods. These solutions include a compliant robot that assist the human in lifting the heavy object, a surveillance and worker assistant system that support the worker during the packaging system and finally, a flexible gripper that enables the grasping of objects of different shape and size.**

I. Introduction

The handling and manual packaging of heavy goods is an exhausting task for human workers. Therefore, it is desirable to develop solutions for supporting the manipulation of these goods. This can be done with gravity compensating handling aids or even by an automatic manipulation system. Such a system should not only be capable to manipulate heavy goods safely also in the presence of human workers, but also be flexible in a way to be easily adaptable to handle new or similar goods. Furthermore, the system should support the worker as good as possible, which includes also assistance for inexperienced users.

The close linkage of human and machine in cooperative production tasks should make use of the strengths of both sides. Typically, an automated assembly system provides a couple of advantages such as operation without breaks and fatigue and high productivity for simple assembly tasks. Today, especially for the assembly or packaging of heavy or bulky parts, weight compensators/balancers are used. Since these systems do not compensate for inertial forces, even small mistakes lead to work-related injuries (lower back pain, spine injuries). According to the statistics of the Occupational Safety & Health Department (OSHA) of the US Department of Labor [1], more than 30% of the European manufacturing workers are affected by lower back pain which brings in enormous social and economic costs.

The paper is organized as follows: Section II gives an overview of different human-robot assistance systems and packaging systems that are available in research and industry. In section III, different technologies that are developed for a flexible packaging cell are introduced and described how they can support the human worker during the packaging process. Section IV provides information about process improvements that the developed components can provide to the human worker and finally section V summarizes and concludes the paper.

II. State-of-the-Art

A. Assistance Systems

Today's human-robot collaboration is affected by a separation of the human worker and the robot. The worker programs the robot or controls it from a distance using a teach-in panel. This allows for offline and static tasks to be executed. To ensure safety, the workspaces of humans and robots are strictly separated either in time or space [2]. In the automotive industry for instance, human workers are completely excluded from the production lines where robots execute assembly steps. Industrial robots are not yet integrated along human workers in assembly line manufacturing.

In general, co-operation between humans and industrial robots is emerging more and more in many research areas. Main topics in these research areas are human-machine interaction as well as safety requirements, issues and union decrees [3]. In some industrial applications, mobile platforms with a mounted industrial arm are used for robot co-operation (e.g. welding processes) [4]. For the handling of heavy work pieces in an automated production environment, concepts of human-robot co-operation are developed, in which an industrial robot hands over heavy loads (e.g. rear axle gear unit) to the worker [5]. The robot Baxter [6] is also designed to help in manipulation tasks and it can safely interact with humans and can perform repetitive production tasks very easily, but it is not capable of lifting very heavy goods. In the following a short overview about related assistance systems and packaging approaches is presented. Krüger et al. [7] provide a very exact overview of the state of the art in human-robot collaboration.

Industrial applications of human-machine co-operation are mainly to be found in automotive industry. Intelligent automation devices (IAD) which supports the worker in

*Research supported by European Union.

Martijn Rooker, Alfred Angerer and Andreas Pichler are with PROFACTOR GmbH, Im Stadtgut A2, 4407 Steyr-Gleink, Austria (+43 (0)7252 885-313; fax: +43 (0)7252 885-101; email: {martijn.rooker, alfred.angerer, andreas.pichler}@profactor.at).

Frank Wallhoff is with Jade Hochschule, Germany (e-mail: frank.wallhoff@jade-hs.de).

Jürgen Blume and Alexander Bannat are with Technische Universität München, Germany (e-mail: {blume, bannat}@tum.de)

Paolo Ferrara is with Ferrobotics Compliant Robot Technology GmbH, Austria (paolo.ferrara@ferrobotics.at).

Aitor Olara is with Tekniker, Spain (e-mail: aitor.olara@tekniker.es).

Janne Kiirikki is with VTT Technical Research Centre of Finland (e-mail: janne.kiirikki@vtt.fi).

assembly tasks such as axle sequencing, cardboard blank handling, engine block handling, and transmission sequencing have been introduced in the industry. This IAD technology was originally developed by the US Company Cobotics in 2003 and is based on fundamental research done by Colgate and Peshkin [8]. Further research for so-called amplifying assist devices (IPAD) was done by Fraunhofer IPK [9].

Finally, augmented reality (AR) technology is studied in many research projects for usage in industrial applications as an assistance system. In [10], the AugAsse system is developed through which a worker can see additional graphical information superimposed on his/her view of the real world. Graphical instructions, text/symbols and virtual objects are used for advising the worker with the assembly tasks. In [11] gestural interaction functionalities were implemented to support hands-free walk-through in assembly instructions. In [12] a multi-modal virtual and augmented reality assistance system is created for improving training skills of workers in maintenance and assembly tasks. Similarly, in [13] a user is provided with augmented reality instructions for maintenance of a printer. Further work conducted in [14] is applying augmented reality for user training, programming and operation, and service and maintenance of an industrial robot.

B. *Packaging*

Packaging has been a huge part of the production industry. Many products are delivered to the customer in a packaged form. Various packaging approaches are applied in the industry nowadays. In many companies the human workforce is still very large, especially in low wage countries. People are standing along conveyor belts and placing the objects inside their package. As this work is very tedious and in many situations also not very ergonomic, more and more companies are moving towards automated packaging. In 2010, the number of robots sold worldwide for packaging increased considerably from 232 units to 653 units, accounting for a share of 4% of the total supply of robotics units sold [15]. Nowadays, almost all robotic suppliers are providing their own solutions for robotized packaging. One of the most well-known packaging systems is the FlexPicker™ developed by ABB [16].

The CustomPacker project [17] develops a customizable and flexible packaging cell for mid- to uppersized electronic consumer goods using industrial robots. The goal is to team humans and industrial robots, combining both the skills of the robot and the human worker in order to box large and heavy consumer goods. Furthermore, introducing a universal robot assisted packaging cell, the number of packaging cells can be reduced, while simultaneously allowing for boxing of different product variants without huge reconfiguration costs.

III. Assistive Technologies

Within the CustomPacker project many different technologies are developed that assist the human worker during the packaging process. The goal is to improve the packaging process on the levels of throughput, but especially on creating green work places, alleviating the human worker from stressful, repetitive and dangerous situations. In the following sections some of the assistant technologies will be introduced. Furthermore it will be clarified how the technologies can be of advantage to the human worker.

A. *Robot Assistant System*

Within the concept of the CustomPacker packaging cell, an industrial robot needs to work together with a human worker in the same work place safely, efficiently and interactively. Various compliant robots are already available on the market like the UR10 [18] from Universal Robots A/S, which is capable of handling up to 10 kg, or the KR 5 SI [19] from MRK Systeme GmbH, which is DIN EN ISO 13849 certified and can handle up to 5 kg. Unfortunately, electronic consumer goods can reach weights of up to 30kg or more, therefore a new compliant robot and control has been developed within the project.

1) *Compliant Robot*

The compliant robot comprises of a vertical elevation axis, a compliant modular arm operating horizontally (like a “scara” arm) and a “soft” actor/sensor element called active contact flange. Figure 1 depicts the modular compliant arms with the contact flange. All components are built together in a modular way, and designed in different sizes, to cope with different customer needs.

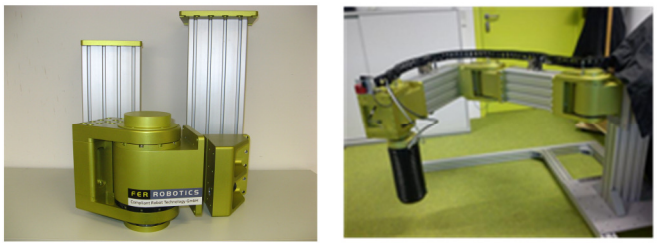


Figure 1: Modular compliant arm (without vertical axis)

Special focus was given to safety needs due to the planned human robot collaboration. The goal to raise rather heavy loads still being safe, led to the compromise of dividing movements into vertical (dangerous) and horizontal (safe) ones. Vertical forces are high; therefore this movement has to be done, where the human is not present (almost in automatic mode). Horizontal forces were held low, to guarantee an intrinsic safety. It was also decided to make the robot “switchable” to a passive mode, where the human can use it like a manipulator or as a force feedback element. Here safety is also intrinsic.

The horizontal movement is driven by very low-power motors. Due to the scara kinematic, it is only needed to overcome inertia, friction in bearing and gears, but not the weight, which is held by the vertical structure. This provides already a basic safety. The joints are also provided with a viscoelastic coupling. This allows moving the arm slightly, even if the breaks are activated, by pushing it with an external force.

Starting from such physical measurements, different risk-minimizing measures can be induced. In case of a collision an overload can be detected comparing the displacement of the coupling with the motor moments arising from mechanic and dynamic calculations to follow the predefined collision free path. In case of a detected collision, drives can be switched off or braked.

Thus if a person should be clamped by the machine, the elasticity in the joints allows a self-liberation adding further safety.

To further mechanically reinforce the flange and reduce risks, an external parallel mechanism was applied (see Figure 2). The mechanical springs compensate the gripper weight. Therefore not the whole payload of the gripper will act on the workers hands, but only the weight of the object that is being handled (even in case of software failure of the flange).

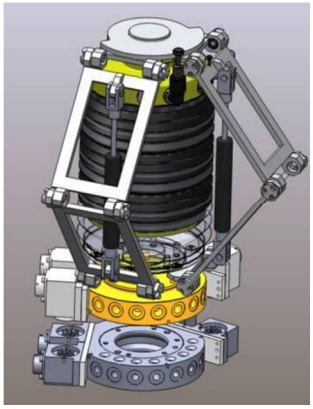


Figure 2: Mechanical flange reinforcement

2) Path Planning

The manipulation planner creates the collision-free path needed for the robot to move its arm to the object that needs to be grasped. The detection of the object to be grasped is described in a later section. The planner is further responsible for creating the path needed for simulating the robot movements. The calculated path is also translated into a robot-specific programming language.

Before a detected object can be grasped it needs to be prepared, which happens offline. In this preparation phase, grasping points are positioned on the CAD model of the object. Based on the identified grasping points, the planner collects all possible grip poses of the recognized object. Using an Inverse Kinematics (IK) solver all the grasping points are considered for a collision free path. This is done by checking the geometry of the robot with the attached gripper to all static geometries of the robot work cell by using simplified CAD models. The following checks are performed for collision-free paths:

- Verification if the IK solution of the grip pose is collision-free;
- Determining if the direct path from the home position to a fixed distance above a grip pose, the so-called pre-grip point (PGP), is collision-free;
- Determining if the direct path from the PGP to the grasp point is collision-free;
- In the last step, the planner also checks if the reverse path (same as the grasping point, only vice versa) is collision-free. Here the difference is that the grasped object is now dynamically attached to the gripper.

When a collision-free path is found, the path planning algorithm writes a robot-specific movement file, which is then executed by the robot. In the CustomPacker project the robot is a compliant SCARA robot. The reachability is very constrained, because the gripper cannot have a transformation with an x- or y-rotation (with respect to the robot base). The planner aligns a tilted xy plane to the xy play of the robot (if this plane is not extremely tilted) before doing an inverse kinematic on the transformation of the grip or pregrip point.

B. Worker's Surveillance and Assistant System

The assistance system is designed to help the human worker fulfill his dedicated task for new product variants to be manipulated.

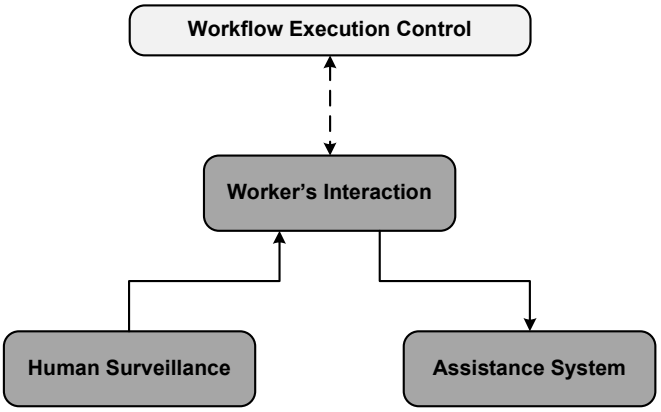


Figure 3: Worker's Assistant consists of two main modules

1) Worker's Surveillance

Although the robot system is intrinsically safe, it should still avoid collisions that may occur between the system and the human worker. A surveillance system is integrated within the workcell, which consists of a camera system (Microsoft Kinect) and a capacitive sensor mat, which is installed on the floor of the cell. These devices enable the workcell to identify the position of the human worker.

The Kinect sensor produces a wireframe model of the human body. It directly calculates the position of the tracked body in world coordinates. By calibrating the cell coordinate system with the Kinect coordinate system, the position of the worker becomes available from the Kinect data.

The capacitive sensor mat gives sensor data when objects are moving or standing on the shop floor. Since the sensors on the mat have a specific pattern, the position of the data producing objects can be directly calculated. Together with the Kinect data, the surveillance system is capable of delivering estimates about the workers position within the cell.

Based on the worker position data, the activity of the human worker can be estimated by using the available workflow information about the packaging process. With a rule-based approach, relevant positions are combined with the workflow information. These rules are used as event-triggers for estimating the current worker action. For example, if required packaging material is stored at a specific location, the worker has to fetch this material. By entering and afterwards leaving this area, the surveillance logic produces the event *fetch item* from this observation. The workflow execution control can react on this event and trigger the next step.

2) Worker Assistance System

The assistance system is designed to help the worker fulfill his dedicated tasks for new product variants to be manipulated. The system is equipped with two modalities for supporting the worker. It features a visual and an acoustic information channel. A projection unit mounted above the worker's cell uses projected light patterns to guide the worker to important areas in the cell or highlight regions relevant to perform the next step.

The assistance system breaks the packaging process down into elementary steps that the worker has to perform, using different auxiliary devices. Packaging processes will be represented in a form of symbolic actions sequences. The product that needs to be packed is recognized by the system in the production line. The packaging instructions are then retrieved from a database using the product ID. The product needs to be identified in advance so that the human worker has sufficient time to prepare the correct packaging box and material for the product before the product arrives at the packaging station. The detailed instructions consist of sequential elementary phases the human worker has to follow. The human worker packs the product correctly and in the planned time using the instructions given by the assistance system.

The worker is monitored through the Kinect sensor and in addition to the worker location, his hand locations are also used to deduce whether he has completed some tasks or not. To assist the worker, the system spotlights the location of the next pick and place operation. Additionally acoustic instructions are given. In Figure 4, the projection of this unit is highlighting the region marked in green light. In this example, the worker has to place the cardboard box on the conveyor belt at the marked spot. Once the worker reaches the region, he also receives further instructions via speech, generated by the assistance system. Trained workers, who are already used to the required process can reduce the level of information presented by the system, or choose to even turn if off completely.

Figure 4: The assistance system is providing instructions and light patterns (green light on the shop floor) to guide the worker.

C. Flexible Gripping Assistance System

All components defined so far support the human worker during the packaging process and increase the ergonomics. The flexible gripping concepts not only support the human worker, but also increase the flexibility of the system and enable the packaging of various electronic consumer goods.

1) Object Pose Recognition

The object pose recognition concept relies on a 3D surface scan of the object to be grasped. Therefore, the software ReconstructMe [20] is used to generate an accurate 3D surface model of the object to be grasped. The ReconstructMe system consists of a depth image input device (e.g. the Microsoft Kinect or Asus Xtion), a computer for the calculation of the 3D surface and visual feedback to the user. For 3D surface reconstruction, the user takes the depth image input device in his hand and films the object from different viewpoints. The 3D surface model is captured in real-time, at approximately 30 fps. The real-time capability is achieved through calculation with the GPU, which provides data parallel execution.

ReconstructMe reconstructs a copy of the surface via a truncated signed distance function (TSDF). TDSF represented by a union of volume grids with a configurable size in both dimension extensions (x, y, z) and grid size. Thus, the 3D reconstruction is limited to the size of the volume, and the accuracy can depend on the grid size. Each depth image frame of the input device is integrated via the TSDF into the volume. Since the camera can move around freely, the position of the camera has to be tracked relative to the volume. Therefore, the previous depth images and the current one are transformed to point clouds in the camera coordinate system and aligned by a high speed version of the Iterative Closest Point (ICP) Algorithm. Once the reconstruction is finished, a triangulated model can be exported using the marching cubes algorithm [21].

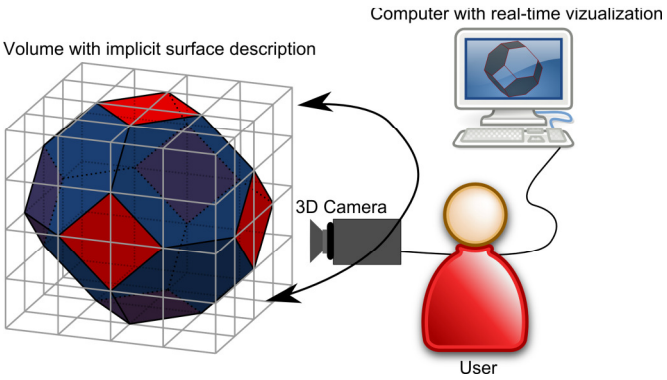


Figure 5: Setup for constructing CAD models of unknown objects with ReconstructMe

Figure 6: Reference scan of the TV-set with ReconstructMe

2) Flexible Gripper

The flexible gripper is developed based on the concept of a hybrid solution between deformable end-effectors and grippers with a high number of DOFs.

On the one hand, three independent linear stages provide adaptability to very different size parts, from under 100 mm to up to 1000 mm. Moreover, these stages enable the gripping in four different directions, so that gripping can be performed from both the outer and the inner side of the parts, as depicted in Figure 8.

Figure 7: Mechanical arrangement of the flexible gripper

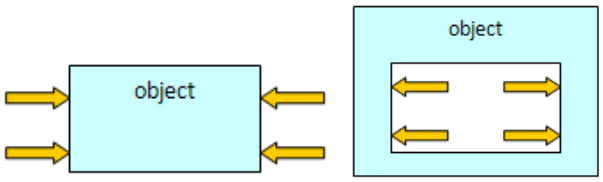


Figure 8: Gripping from outside and from inside

On the other hand, a very compliant contact area implemented passively with an elastomeric pad assures good adaptation to small features of the part.

IV. Performance Evaluation

During the setup of the demonstrator, a particular packaging cell setup was constructed in the lab at the Technical University Munich, to have a first reference value. This is because the final cell layout may be very variable, depending on precise customer requirements. Relevant elements and tasks to be tested and evaluated at subsystem level are:

- The worker recognition (detection, localization, activity recognition, …)
- The worker assistance system
- The 3D vision and path-planning system
- The robot (moving and grasping)

Components were first evaluated separately at subsystem level and then combined to the overall demonstrator, to measure the system performance.

A. Worker recognition

For the evaluation at the component level, the different partner performed first evaluation tests at their own sites to gain insights in the technologies.

Detection of the position (center of gravity) of one worker (general measurement, not in relation to a specific task), with the assistive capacity mat has an accuracy of 250mm and the Kinect Sensor provides a depth image of 10-40mm, which results in a pose estimation accuracy of up to 100mm. Here the following potential error sources have to be taken into account that the position on the capacity mat is highly dependable from the sensor based measurements and that the Kinect is influenced by reflections, infrared light sources and occlusions. The number of recognized events (worker changes from one activity position to the next one) in respect to the total expected events has been evaluated. During the observed packaging cycle, 11 different events (packaging of object, placing of boxes, etc.) have to be recognized. For this evaluation 165 events were generated by one expert user and two novice users:

- Using the capacitive sensing mat > 95% of all events were recognized correctly
- Using Kinect, no restriction in clothing occurred. Direct sunlight might disturb the depth tracking. However, in the presented setup, this was not observed. The Kinect was tested, but finally not chosen for further measurements, due to occlusion in front of the conveyor.

B. Worker Assistance

The worker assistance system has been evaluated so far only by having students learns and complete the work cycle either with giving them paper instructions, or by being guided with the worker assistance system. As the test sample was small (<10), the results are mostly qualitative from notions taken from observing and comments from the would-be-workers. The cycle time with the worker assistance system was significantly slower (around +40% to cycle time), but this can mostly be explained by the fact that the assistance system gave all instructions by vocalizing them, and this took a couple of seconds for each work phase.

In comparison, the written instructions were very short and did not consume much time. This could be different having a greater set of different and more complex activities. The test workers that completed the work cycle with the assistance system seemed to like the system for helping them learn the work cycle, but a general comment was that it is too repetitive to have the system constantly helping them. Further comments were also that the assistance system would be good for packing various products and for processes with variances in the workflow. The next step is to test with real employees from end-users, which will require further integration and teaching material.

C. 3D Object Recognition and Path-Planning

For the standalone object recognition of the part to be grasped, the accuracy has been calculated (without a reference to the robot system). Within the different measurements, a maximum error of 4mm has been measured with respect to the borders of the object to be grasped. Using these results in the path planning measurements, no errors have been detected in the collision-free planning during the test cycles if the position and orientation of the object is relative similar to the reference model. The ranges are about +/- 150 mm translation in all directions and +/- 20° orientation.

D. Gripper and Robot Performance

Several tests have been carried out to check the gripper performance. On the one hand, the available strokes and distances between the fingers have been checked, resulting in 140mm and 820mm strokes for short and long stroke stages respectively. Gripping force measurements have been carried out locking force transducers between the fingers. The obtained results for maximum forces are in the range of 540 +/- 20N range. On the other hand, force control within 3% of the maximum forces has been achieved, which enables fine force control for gripping applications. The reliability of the gripper has been checked by means of long time gripping. Trials of around 5 hours part holding have been successful. Finally, the gripper has performed successfully gripping lightweight parts like a 40x300x200 mm cardboard box, as well as 30kg and 46" TV sets.

Finally, the robot functional parameters are evaluated and provide a gripping precision of 2mm (with a +/- 1mm repeat accuracy). A switch between automatic mode and gravity compensation mode is integrated enabling different interaction modes. Measurements of max speed resulted into velocities of 100mm/sec in the vertical direction and 300mm/sec in a horizontal direction.

V. CONCLUSIONS

The packaging industry is looking more and more at approaches for optimizing the packaging process and providing humans with ergonomics enhancements. This paper introduced different assistant technologies that have been developed within the CustomPacker project to assist the human worker and improve the ergonomics in the workcell. A compliant robot has been developed that supports the human worker during the packing process, where the robot takes care of the lifting of the heavy part. Furthermore, a surveillance and assistant system has been developed that tracks the human and with augmented reality teaches new workers the packaging process. Finally, a flexible gripper with object detection technology is developed that enables the grasping of objects of various weight and shapes.

Concluding it can be mentioned that the functionality of the single modules fulfills the requirements (at demonstration level), although first test regarding the performance of the full demonstration was over the goal (107 instead of re quested 90 sec per cycle). But due to the high remaining time buffer for the human as well as for the robot (human is paused for 37 sec per cycle, robot for 42 sec), the expected results in the industrial setup are very promising.

ACKNOWLEDGMENT

The authors would like to thank all partners of the project for their support. This research project is supported by the European Commission under the 7th Framework Programme through the 'Factories of the Future' action under contract No: PPP-FoF-260065.

REFERENCES

[1] Occupational Safety and Health Administration, [Online]. Available: http://www.osha.eu.int. Last accessed: March 2013

[2] DIN EN 775 (1993): Manipulating industrial robots – Safety. Beuth Verlag GmbH, Berlin, Germany. 1993

[3] Bischoff, R., Schmirgel, V., Suppa, M., „The SME Worker's Third Hand", SME Project. [Online]. Available: http://www.smerobot.org/index.php. Last accessed: March 2013

[4] Hägele, M., Helms, E., "rob@work: der Assistenz der Zukunft - Mobile Assistenzroboter in der industriellen Fertigung", Fraunhofer IPA, [Online]. Available: http://www.care-o-bot.de/english/RobAtWork.php. Last accessed: March 2013

[5] Schraft, R.D., Meyer, C., Parlitz, C., Helms, E., „PowerMate – a safe and intuitive robot assistant for handling and assembly tasks", *IEEE International Conference on Robotics and Automation 2005 (ICRA 2005),* pp. 4085-4090, April 18-22, 2005

[6] Robot Baxter: A unique robot with unique features [Online]. Available: http://www.rethinkrobotics.com/index.php/products/baxter. Last accessed: March 2013

[7] Krüger, J., Lien, T.K., Verl, A., „Cooperation of Human and Machine in Assembly Lines", *CIRP – Annals,* Vol. 58, No. 2, 2009

[8] Colgate, J.E., Wannasuphoprasit, W., Peshkin, M., "Cobots Robots for Collaboration with Human Operator", *Proceedings of the ASME Dynamic Systems and Control Division*, DSC, Vol. 58, 58, pp. 443-440, 1996

[9] Thiemermann, S., Schraft, R.D., „team@work - Mensch-Roboter-Kooperation in der Montage", In: *Automatisierungstechnische Praxis atp: Praxis der Mess-, Steuerungs-, Regelungs- und Informationstechnik 45,* Nr. 11, pp. 31-35, 2003

[10] Salonen, T., Sääski, J., Hakkarainen, M., Kannetis, T., Perakis, M., Siltanen, S., Potamianos, A., Korkalo, O. and Woodward, C., "Demonstration of assembly work using augmented reality", *ACM International Conference on Image and Video Retrieval (CIVR2007)*, Amsterdam, The Netherlands, July 9-11, 2007

[11] Siltanen, S., Woodward, C., Valli, S., Honkamaa, P. and Rauber, A., "User Interaction for Mobile Devices", In: P. Maragos et al. (Eds.). *Multimodal Processing and Interaction*. Springer, 2008, pp. 314-318

[12] Gutiérrez, Teresa; Gavish, Nirit; Webel, Sabine; Rodríguez, Jorge; Tecchia, Franco in Bergamasco, Massimo (Ed.): Skill training in multimodal virtual environments, Boca Raton/Fla.: Taylor & Francis, 2013 (Human Factors and Ergonomics), ISBN: 978-1-4398-7895-8, p.227-239

[13] Steven Feiner, Blair Macintyre, and Dorée Seligmann. 1993. Knowledge-based augmented reality. Commun. ACM 36, 7 (July 1993), 53-62. DOI=10.1145/159544.159587 http://doi.acm.org/10.1145/159544.159587

[14] Bischoff, R.; Kazi, A., "Perspectives on augmented reality based human-robot interaction with industrial robots," Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on , vol.4, no., pp.3226,3231 vol.4, 28 Sept.-2 Oct. 2004 doi: 10.1109/IROS.2004.1389914 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1389914&isnumber=30278

[15] World Robotics 2011 Industrial Robots, ISBN 978-3-8163-0635-1, published by the IFR Statistical Department, hosted by VDMA Robotics + Automation, Germany

[16] ABB FlexPickerTM, Access Date September 2012. [Online]. Available: http://www.abb.com/product/seitp327/cf1b0a0847a71711c12573f40037d5cf.aspx. Last accessed: March 2013

[17] CustomPacker – Highly Customizable and Flexible Packaging Station for mid- to upper sized Electronic Consumer Goods using Industrial Robots. [Online]. Available: http://www.custompacker.eu. Last accessed: March 2013

[18] Universal Robots, "Collaborative Robot Solutions". [Online]. Available: http://www.universal-robots.com/GB/Products.aspx. Last accessed: March 2013

[19] MRK-Systeme GmbH, "KR 5 SI (SafeInteraktion)". [Online]. Available: http://www.mrk-systeme.de/e_produkte_interaction.html. Last accessed: March 2013

[20] ReconstructMe, "Digitalize your world". [Online]. Available: http://www.reconstructme.net. Last accessed: March 2013

[21] Lorensen, W.E., and Cline, H.E., "Marching Cubes: A high resolution 3D surface construction algorithm", In: *Computer Graphics,* Vol. 21, Nr. 4, July 1987

HOTINT - a Free Flexible Multibody System Simulator for Robotics Applications

Martin Saxinger[1], Peter G. Gruber[1] and Johannes Gerstmayr[1]

Abstract— **In this article the multibody dynamics simulator HOTINT is presented, with particular emphasis on its features for dynamics simulation of robots. Many freely available tools for robotics simulations are based on the idealized kinematics of the robot and a minimal set of coordinates is utilized for the description of the motion. Thus, joints as well as links are usually modeled as rigid bodies. HOTINT is based on a redundant coordinate multibody system dynamics formulation, which allows to model each link as rigid or flexible body and to interconnect the bodies with rigid or flexible links. The underlying formulation is not based on minimal coordinates, which could model the joint angles and translations directly. Thus, powerful actuators and sensors are available in order to make these coordinates accessible. Finally, the control and power electronics part of the robot can be defined by means of port-blocks, which connect sensors, actuators and loads. During the solution process, the whole scenery is visualized in 3D during simulation, which allows close-to-realtime applications.**

I. INTRODUCTION

In order to study and optimize the performance of nowadays high-speed and low-weight robots, flexibility in joints as well of the links needs to be included. A coupled analysis of all parts, including the drive train, the power electronics and the feed-forward and feed-back control algorithms.

Commercial multibody and multiphysics simulation engines are extremely powerful and costly. Furthermore, the complexity of simulation packages requires experts in order to be able to model complex mechatronics systems. An automated model generation and evaluation procedure (e.g. driven by an external optimization program) is difficult to be set up. A drawback of commercial software especially designed for robotics simulation, is that most codes are designed towards kinematic analysis, contact detection, energy consumption, path planning, etc., but not regarding deformable links, flexible joints or more advanced mechatronical components. the Virtual Robot Experimentation Platform (V-REP) [1], which is free for academic purposes, the Microsoft Robotics Developer Studio [2] and others.

In robotics simulation, there are many freely available tools for specific tasks of the simulation. Many of these codes are either an extension of physics engines for graphical visualization or based on standard kinematics or minimal coordinate dynamics formulations. The latter approaches cannot be extended in a straightforward manner to model more complex behavior, e.g. flexible joints or a sliding joint along a deformable body. Examples of free software tools are e.g. dVC3d, ORCA-Sim, SimRobot, OpenHRP3.

In this paper, the simulation of the dynamic behavior of robots shall be based on flexible multibody dynamics simulation. The C++ multibody dynamics code HOTINT has been designed for this purpose and various realistic robots have been simulated [3], [4]. As a main feature - in comparison to many other available freeware - a complete reference manual of each object is provided for the user. The purpose of this approach is

- understanding the influence of flexibilities of joints and links [3]
- studying the influence of friction and clearance [5]
- automated identification of unknown parameters [4]
- reduction of vibrations due to appropriate path planning and feedback control strategies

The current version V1.1 includes components such as point mass, rigid bodies, flexible (large deformation) beam elements, complex point-based joints, classical mechanical joints, port-blocks for mechatronics applications and many other features such as loads, sensors and graphical objects. Furthermore, 3D (online) graphical visualization is integrated and helps to reduce modeling errors.

The user can create and modify a model in the multibody system code either by means of a text input script file or directly in the graphical user interface. The script-language of the text input file is designed for simple manual model definition and is sketched within the present paper. A full documentation is available, including a reference manual to each object and the usage of the script language.

HOTINT is currently released under the permissive free BSD 3-clause software license. It is planned to include further important features in the near future.

A short overview on the structure of the code is given in Sec. II. In Sec. III particular emphasis is given on how to model robotics applications. A a short example model (written in the script language) is outlined in Sec. IV.

II. STRUCTURE OF THE CODE

The core part of HOTINT is based on a *High Order Time INTegrator*, which is a time integration library for differential algebraic and discontinuous equations. Now, it should serve as a versatile simulation tool for complex mechanical and mechatronical applications throughout numerous industrial and scientific projects [6]. Particularly applications in robotics and control (see, e.g., [5], [3], [7], [8]) have shaped the simulation library over the years. In the following section, a short overview on the structure of

[1]All authors are with Mechanics & Model Based Control, Austrian Center of Competence in Mechatronics (ACCM), Linz, 4040, Austria, Email: martin.saxinger, peter.gruber, johannes.gerstmayr at lcm.at

the freeware is presented. For more detailed information the interested reader is referred to the complete reference manual [9], which can be downloaded together with the software at: `www.hotint.org`

The software consists of several independent core modules, each of which are implemented in separate C++ libraries. The most significant of those are summarized in the following paragraphs:

a) Multibody system (MBS) kernel: links objects and performs assembling of the global system; handles options and solver settings; calls and interacts with the numerical solvers (cf. Sec. II-B).

b) Object library: contains all available objects (cf. Sec. II-C).

c) Numerical Solvers: are provided for six different computation modes (cf. Sec. II-D).

d) Mathematical libraries: covers basic linear algebra (dense and sparse) functionality and data structure.

e) Graphical user interface (GUI): manages all communication with the user; provides a 3D-visualization via an OpenGL front end; offers tools for model creation and manipulation, and a plot tool for sensor values, see Fig. 2.

f) Script Language: enables enhanced model programming, and provides a hierarchical data structure for parameters and solver settings (see Sec. III-A).

A. Equation structure of HOTINT

The equations of motion for mixed 1st, 2nd order and algebraic equations can be written in semi-implicit form as

$$\begin{aligned} \mathbf{M}(\mathbf{u},\mathbf{v},\mathbf{x},\mathbf{z})\,\dot{\mathbf{u}} &= \mathbf{F}_2(\mathbf{u},\mathbf{v},\mathbf{x},\mathbf{z}) \\ \dot{\mathbf{x}} &= \mathbf{F}_1(\mathbf{u},\mathbf{v},\mathbf{x},\mathbf{z}) \\ \mathbf{0} &= \mathbf{G}(\mathbf{u},\mathbf{v},\mathbf{x},\mathbf{z}) \end{aligned} \tag{1}$$

with $\mathbf{u}$ as the vector of generalized coordinates for 2nd order ODEs (e.g. mechanics). For more details see [10]. The vector $\mathbf{x}$ represents the generalized coordinates for 1st order ODEs (e.g. hydraulics, control,...) and $\mathbf{z}$ are the algebraic variables of the system. The variable $\mathbf{v}$ is a acronym for $\dot{\mathbf{u}}$. $\mathbf{M}$ denotes the mass matrix (possibly nonlinear) and the vector $\mathbf{F}_2$ denotes the right-hand side of the 2nd order equations. $\mathbf{F}_2$ includes elastic and external forces as well as damping, gyroscopic terms and Lagrange multipliers. The right-hand side of the first order equation is represented by $\mathbf{F}_1$ and $\mathbf{G}$ denotes the vector of all kinds of algebraic (constraint) equations.

B. Multibody system kernel

Every object of a multibody system, see Sec. II-C and Fig. 1, adds a certain set of their own (local) equations to the whole set of (global) equations. The crucial task of the *Multibody System kernel* is to assemble these global equations based on the connections of the multibody system, and to provide the system equations to the solver. Apart from that, the kernel is responsible for setting up the model, steering the simulation, organizing in- and output of file data, as well as accessing or modifying specific element data.

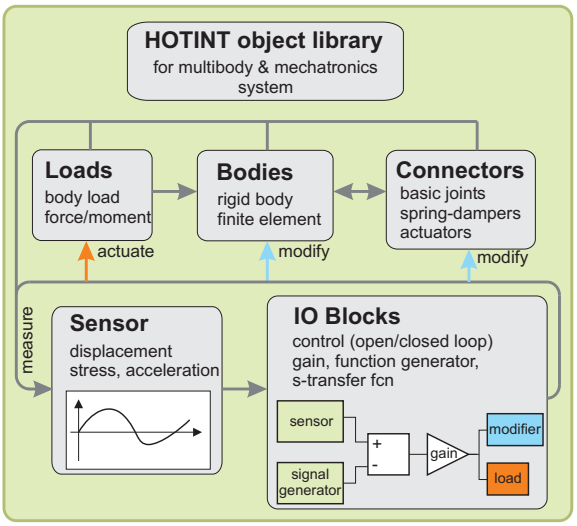


Fig. 1. Structure of the multibody system (MBS).

C. Object library

The object library provides a set of rigid bodies (links), basic joints, loads and sensors, similar to any robotics simulation code. As a main feature of HOTINT, there exists a variety of flexible bodies (Finite Elements), connectors (actuators, springs, and dampers), loads, sensors, and IO-Blocks (controllers) – as outlined in Fig. 1.

Among flexible bodies are structural Finite Elements for beams, available either in geometrically exact formulation (for large deformation processes, ropes, cables, etc.), or in linearized form (faster). Joints are designed such that complex combinations of bodies and joints are possible, e.g. a point of body may move along a deformable body's axis (sliding joint). Since also *flexible* bodies are available, the object definition is based on generalized (redundant) coordinates for the bodies. Joints reduce the number of coordinates, which is different from many robotics simulators which apply joint based formulations, due to the assumption of purely *rigid* body dynamics.

In the core part of the code, objects are represented by means of first order and second order differential equations, algebraic equations and jump or switching conditions. Furthermore, the objects include standardized coupling conditions (for joints and loads), graphical representation and measurable quantities (for sensors). These objects define bodies (links) and joints and may be easily extended. The precise underlying equations can be found in a couple of papers [6], [10] and are not presented in this paper.

D. Numerical solver

The multibody system is directly coupled with the nonlinear static and dynamics solvers, which are designed to solve complex dynamical problems based on first/second order ordinary differential equations (ODEs), differential algebraic equations (DAEs) up to index 3, and jump conditions.

There are three basic computation modes:

- static computation (initial equilibrium),
- dynamic computation (forward simulation),
- eigenmode computation,

For each mode, it is possible to perform

- parameter variation,
- sensitivity analysis, or
- optimization / parameter identification.

D.1. Forward simulation

The 2nd order part of (1) is transformed into a first order system

$$\begin{aligned} \dot{\mathbf{u}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= \mathbf{M}^{-1}\mathbf{F}_2(\mathbf{u},\mathbf{v},\mathbf{x},\mathbf{z}) \end{aligned} \tag{2}$$

with the inverse of the mass matrix $\mathbf{M}$. Additional variables $\mathbf{y}$ are used e.g. for switching, history, time delays, etc.. DAEs (1) and (2) are solved by means of generalized Runge Kutta tableaus either directly with index 3 (RadauIIA methods) or reduced to index 2 (solvable e.g. by trapezoidal rule). The numerical drift off in index 2 formulation is in many cases neglectable, or stabilized. The Runge-Kutta methods written in 'K-form' with the unknown vectors $\mathbf{K}_{iv}$, $\mathbf{K}_{ix}$ and $\mathbf{z}_i$ leads to a nonlinear equation for every stage i.

$$\begin{bmatrix} \mathbf{M_i K_{iv}} \\ \mathbf{K_{ix}} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{F_2}\left(\mathbf{u_i},\mathbf{v_i},\mathbf{x_i},\mathbf{z_i}\right) \\ \mathbf{F_1}\left(\mathbf{u_i},\mathbf{v_i},\mathbf{x_i},\mathbf{z_i}\right) \\ \mathbf{G}\left(\mathbf{u_i},\mathbf{v_i},\mathbf{x_i},\mathbf{z_i}\right) \end{bmatrix}, \tag{3}$$

The formulation in (3) is specially designed for flexible MBS (or robots) with many DOF and can be solved efficiently in contract to (2). Constitute of several stages:

$$\mathbf{v}_i = \mathbf{v}_0 + \tau \sum_{j=1}^{n} A_{ij}\mathbf{K}_{jv} \tag{4}$$

$$\mathbf{u}_i = \mathbf{u}_0 + \tau \sum_{j=1}^{n} A_{ij}\mathbf{K}_{ju} \tag{5}$$

$$\mathbf{x}_i = \mathbf{x}_0 + \tau \sum_{j=1}^{n} A_{ij}\mathbf{K}_{jx} \tag{6}$$

Each step is solved with Newton's method (modified or full Newton). The evaluation step does not require the factorization of the mass matrix, see [10]. Discontinuous events (such as switching, contact, plasticity, etc.) are solved by an additional iteration outside the Newton's method. For this purpose the additional history variables $\mathbf{y}$ are used.

A large number of implicit Runge Kutta tableaus are available for the time integration. Fast standard methods like implicit Euler, midpoint and trapezoidal rule are available as well as high order Gauss, Radau or Lobatto schemes. For details on each particular computation mode and solver options, see the reference manual [9].

III. HOW TO MODEL A ROBOTICS SYSTEM

It is recommended to model the robotics system within the intuitive script language. The script language can handle variables and a detailed reference manual is provided. Simple models can be set up directly in the HOTINT GUI (which currently does not support all ways to setup models).

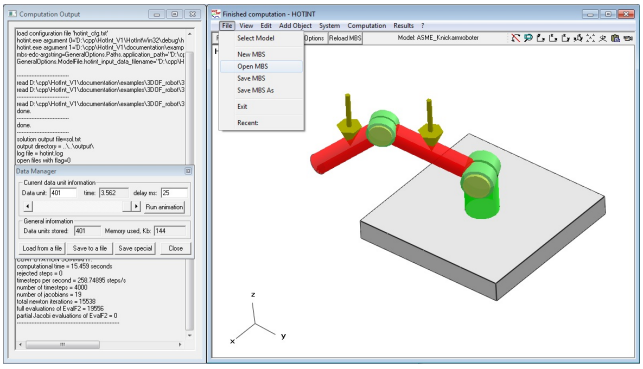


Fig. 2. Model loaded from a model file written in HOTINT script language.

A. Script language and model definition

The HOTINT script language supports a variety of commands for setting up a multibody system, which allows to combine objects like rigid and flexible solid bodies, materials, nodes, constraints, loads, and sensors (see Fig. 1) each of which comes with its specific set of options and initialization parameters.

As compared to some alternative language definition (XML), the script language offers programming facilities (variables, mathematical operations, functions) such that complex systems can be programmed in a user-friendly manner. A sketch of using the script language for setting up a model can be seen in Sec. IV, the full code as well as a detailed reference manual on all commands and specification of objects can be downloaded from [9].

The multibody system model can be defined in and loaded from the *HOTINT input data file* (.hid). As soon as the button *Start* in the tool bar is pressed, the numerical solver is launched with the specific solver settings of the model.

B. Multibody system for robot description

In standard robotics codes, the robot is usually setup by means of its kinematic chain. In this way, more complex joints[1] are difficult to be realized and usually, closed loop chains can lead to problems in modeling and simulation. As an advantage, the joint degrees of freedoms (e.g. relative joint axis rotation) can be actuated and measured in a straight forward manner.

In HOTINT, a robot is constructed by means of bodies, which are defined in reference configuration. These bodies are connected by means of joints (connectors). A challenging task in such a formulation is the joint actuation (forces and torques) as well as the joint angle measurement by means of sensors, which has been included in the formalism consequently. As an advantage, the connection of several bodies can be realized almost without any restriction[2]. Furthermore, an extension to flexible bodies (e.g. beams), which

[1] e.g. with flexibilities for all translations and rotations or joint clearance

[2] as a simple restriction, no redundant constraints may be used in the current version of HOTINT.

increasingly play an important role in robotics simulations, is straight forward.

Joint actuation is more complex as compared to kinematics simulation, because a constraint directly acting on joint displacements or rotations is not recommendable in dynamics simulations (leads to algebraic constraints of index 3). However, joint actuation can be done via an *IOElement-DataModifier* (Fig. 3), see below. In order to model more complex actuators, various parameters of joints and actuators can be accessed with those element data modifiers - e.g. for nonlinear behavior. In order to obtain a feedback control loop, the position, velocity, rotation, etc. is measured with sensors and IO blocks, see Fig. 1, are used to simulate complex controllers for joint torques or forces.

IV. EXAMPLE: FLEXIBLE LINEAR ROBOT

In this section we present how to define the model of a simple flexible linear robot via the script language. Representative parts of the input file in script language are provided in the following (the full model can be downloaded within a set of other examples from `www.hotint.org`). The graphic scenery is displayed in Fig. 4. The basic construction consists of four flexible beams (*LinearBeam3D*). Another beam (the vertical one, which is connected to the picker) and all picker elements are realized as rigid bodies (*Rigid3D*). The set up of a rigid body is very simple: position, orientation, mass and moments of inertia completely define this element. To set up a flexible beam element two special nodes (*Node3DRxyz*) and a material (*Beam3DProperties*) are necessary. A *Rigid3D* has seven DOF, three for translation and four for rotation (although these four variables are not totally independent), and they are stored in the element. A difference of the *LinearBeam3D* compared to the *Rigid3D* is, that the beam has no own DOF. This element is linked with the DOF of the nodes (each node has six DOF) which describe the node displacements and rotations w.r.t the global coordinate system. A advantage of node DOF is that neighboring *LinearBeam3D* elements which use the same nodes are automatically constrained without the need to define a additional constraints. Due to the nodes the beam orientation and end points are defined. Other properties like the cross section size, inertia and flexibility parameters are set in the material. See the short excerpt below how to create these elements in script language.

```
%================================================
%example code begin (excerpt elements):
%
%flexible linear robot
%download full example at: www.hotint.org
%================================================

%load ground and robot dimensions (all variables
%in this excerpt are loaded from parameter file)
Include("flexible_linear_robot_params.hid")

%================================================
%create left column beam: for this purpose the
%definition of 2 nodes and beam material is
%necessary

%beam material (dynamics, elasticity,...)
material_beam_1
{
  material_type="Beam3DProperties"
  name="material_beam_1"
  cross_section_type=1 % rectangular cross section
  cross_section_size=[th_beam_1,th_beam_1] %m
  density=density %kg/m^3
  EA=EA %N
  EIy=EIy %Nm^2
  EIz=EIz %Nm^2
  GAky=GAky %N
  GAkz=GAkz %N
  GJkx=GJkx %Nm^2
  RhoA=RhoA %kg/m
  RhoIx=RhoIx %kg*m
  RhoIy=RhoIy %kg*m
  RhoIz=RhoIz %kg*m
}
nMaterialBeam1=AddBeamProperties(material_beam_1)

%two nodes at the beam ends (the following node type
%contains position, orientation and 6 DOFs, see docu.)
node_1
{
  node_type="Node3DRxyz" %special node type with
                         %global DOFs
  name="node_1" %name in HOTINT element list
  Geometry
  {
    reference_position=[-l_beam_2/2,0,0]
    %position of node w.r.t. global coordinate system
    reference_rot_angles=[0,0,Pi/2]
    %orientation of beam defined by bryant angles
  }
}
n1=AddNode(node_1)

node_2
{
  node_type="Node3DRxyz"
  name="node_2"
  Geometry
  {
    reference_position=[-l_beam_2/2,l_beam_1,0]
  }
}
n2=AddNode(node_2)

%create left column beam (given material & nodes)
beam_1_left
{
  element_type="LinearBeam3D"
  name="beam_1_left"
  Physics
  {
  material_number=nMaterialBeam1
  }
  Geometry.node_1=n1
  Geometry.node_2=n2
}
nBeam1Left=AddElement(beam_1_left)

%================================================
%rigid body (vertical beam, not part of gripper)
rigid_body_1
{
  element_type="Rigid3D"
  name="rigid_body_1"
  Physics
  {
   mass=m4 %kg
   moment_of_inertia=[Ixx4,0.,0.;
                      0.,Iyy4,0.;
                      0.,0.,Izz4] %kg*m^2
  }
```

```
  Graphics
  {
    RGB_color=[0., 0., 1.]  %[r,g,b], range=0..1
    show_element=1  %1=visible element
    body_dimensions=[l_rigid,th_rigid,th_rigid]
  }
  Initialization.initial_position=
  [0,l_beam_1,init_pos_z] %m, local position
  Initialization.initial_rotation=[Pi,Pi,Pi/2]
}
nRigid1=AddElement(rigid_body_1)

%example code end
%=============================================
```

Bodies are connected by rigid joints (*RigidJoint*), sliding joints (*SlidingPrismaticJoint*) and revolute joints (*RevoluteJoint*). Rigid joints are used to produce a stiff connection between two elements or one element and the ground. The user can choose between a Lagrangian formulation and a penalty formulation with stiffness and damping. A sliding joint enables the sliding motion of one body along the local x-axis of another (flexible) body. All other rotations and translations of both bodies are constrained by penalty joint stiffness (with damping). In this example, three sliding joints are used to connect the actuated robot arms. See the script language excerpt for the definition of a *RigidJoint* and a *SlidingPrismaticJoint* below.

```
%=============================================
%example code begin (excerpt joints):
%=============================================

%ground joints - fix left beam to ground
joint_ground
{
  element_type="RigidJoint"
  name="ground_joint_left"

  Graphics.cube_length=1.2*th_beam_1 %m, drawing

  Position1.element_number=nBeam1Left %beam left
  Position1.position=[-l_beam_1/2,0,0]
  %m, local position
  Position2.element_number=0 % 0 denotes ground
  Position2.position=[-l_beam_2/2,0,0]
  %global position
}
nRigidJointGroundLeft=AddConnector(joint_ground)

%=============================================
%sliding joints
sliding_joint_1 % between beam 2 and 3
{
  element_type="SlidingPrismaticJoint"
  name="sliding_prismatic_joint_1"
  Physics
  {
    use_penalty_formulation=1 %1=use k and d
    Penalty
    {
      %rotational stiffness
      k1=stiffness %Nm/rad, (global x-axis)
      k2=stiffness %Nm/rad, (global y-axis)
      k3=stiffness %Nm/rad, (global z-axis)
      %rotational damping
      d1=damping %Nm*s/rad, (global x-axis)
      d2=damping %Nm*s/rad, (global y-axis)
      d3=damping %Nm*s/rad, (global z-axis)
    }
  }
  Geometry
  {
    position_1=[-l_beam_3/2,0,0] %m, loc. pos.
    position_2=[0,0,0] %m, loc. pos.
    element_numbers=[nBeam3,nBeam2]
    %sequence of el. numbers: [n3,n2_1,n2_2,...]
    %beam 3 is sliding on beam 2 (in this
    %example only one element)
    elemind=1
    %start index: n2_1=1, n2_2=2, ...
    %see documentation for more information
  }
}
nSlidingJoint1=AddConnector(sliding_joint_1)

%example code end
%=============================================
```

The path planning of the robot takes place via prescribed joint actuator displacements. For a better understanding see Fig. 3, which shows the procedure of actuation for the open loop control. First of all the desired path is defined via *IOMathFunction*. This element computes the spring length for a certain point of time. A so called *IOElementDataModifier* changes this neutral spring length of the appropriate spring damper actuator. This leads to an elastic drive train coupled to the prescribed displacement of the robot arm. The elasticity and damping of the drive train is tunable by *SpringDamperActuator* stiffness and damping parameters. The given trajectories of the beams, which are not jerk-free, lead to considerable and coupled vibrations of the moving arms. For the path planning part see the script language file.

```
%=============================================
%example code begin (excerpt actuation):
%=============================================

% spring damper act. - elastic gear / drivetrain
spring_damper_actuator_1
{
  element_type="SpringDamperActuator"
  name="actuator_1"
  Physics
  {
  actor_force=0 %no const. actor force added
  forcemode=0 %0=IOElementDataModifier,...
  Linear.spring_stiffness=stiffness_SDA
  Linear.damping=damping_SDA
  }

  Graphics.show_connector=0 %0=invisible

  Position1.element_number=nBeam2 %element
  Position1.position=[-l_beam_2/2,0,0]
  %m, local position

  Position2.element_number=nBeam3 %element
  Position2.position=[-l_beam_3/2,0,0]
  %m, local position
}
nSpringDamperActuator1=
AddConnector(spring_damper_actuator_1)

%example code end
%=============================================
```

A field-variable element sensor (*FVElementSensor*) measures the *y*-displacement of the endpoint of beam 3, which is plotted in Fig. 5. For the generation of the diagram, the own plot tool of HOTINT was used. The following excerpt shows the sensor definition.

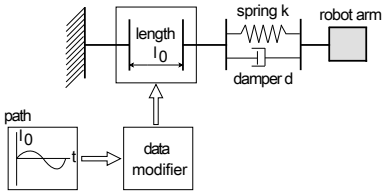


Fig. 3. Actuation of robot arm with open loop control.

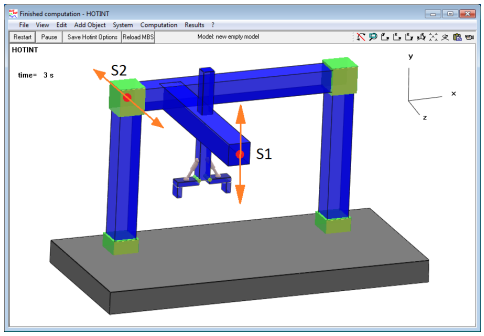


Fig. 4. Example of a flexible linear robot with sliding joints and path planning. Sensor S1 measures vertical vibrations, S2 horizontal vibrations.

```
%================================================
%example code begin (excerpt sensors):
%================================================

%force sensors - force of actuator is not
%position dependent
force_sensor_1
{
  sensor_type="ElementSensor"
  name="force_sensor_1"
  element_number=nSpringDamperActuator1
  value="Connector.constraint_acting_force"
}
nForceSens1=AddSensor(force_sensor_1)

%================================================
%displacement sensors - displacement is a
%field variable (depends on local el. position)
PosYSensor
{
  sensor_type="FVElementSensor"
  name="y-displacement_of_beam_3"
  element_number=nBeam3
  local_position=[l_beam_3/2,0,0] %m, loc. pos.
  field_variable="displacement"
  component="y" %global y-direction
}
AddSensor(PosYSensor)

%example code end
%================================================
```

V. CONCLUSION

In this work, the free flexible multibody system simulator HOTINT is shortly presented, with particular emphasis on robotics applications. A full HOTINT documentation consisting of a user and a reference manual is available for download via www.hotint.org [9]. It provides general information on the underlying concepts, the multibody formulation, and the solver, furthermore includes a documentation of the simulation software HOTINT itself, and contains detailed information about a wide range of elements, constraints, loads, sensors, commands, and options. Advanced components such as 2D/3D contact models, model reduction (component mode synthesis), fluid-structure interaction are currently not available via the user interface but will be available in future versions.

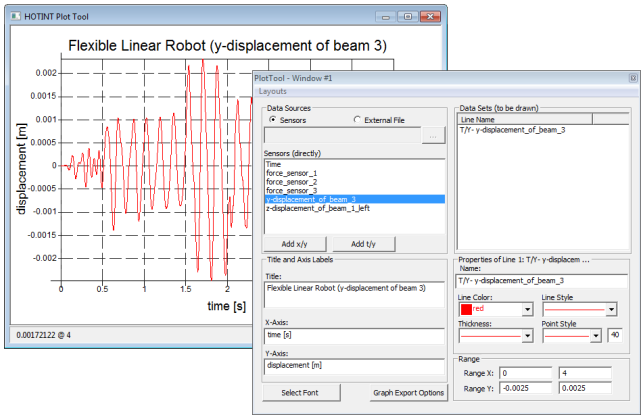


Fig. 5. Vibrations in y-direction, recorded by sensor S1 (see also Fig. 4).

ACKNOWLEDGMENT

The authors gratefully acknowledge support by the the Comet K2 Austrian Center of Competence in Mechatronics (ACCM).

REFERENCES

[1] "v-rep - A virtual robot experimentation platform." [Online]. Available: http://coppeliarobotics.com/

[2] "Microsoft robotics Developer Studio." [Online]. Available: http://www.microsoft.com/robotics/

[3] R. Ludwig, J. Gerstmayr, C. Augdopler, and C. Mittermayer, "Realistic robot simulation with flexible components," in *Proceedings of the Ciras 2008, Fifth International Conference on Computational Intelligence, Robotics and Autonomous Systems, Linz, Austria, 1921 June 2008*, 2008, paper ID 41.

[4] R. Ludwig and J. Gerstmayr, "Automatic parameter identification for mechatronic systems," in *Multibody System Dynamics, Robotics and Control: Proceedings of a Workshop held in Linz 2011*. Vienna: Springer, 12 2012.

[5] R. Ludwig, J. Gerstmayr, C. Augdopler, and C. Mittermayer, "Flexible robot with clearance," in *Proceedings of the 4th European Conference on Structural Control (4ECSC), 8-12 September 2008, St. Petersburg*, 2008, paper ID 221.

[6] J. Gerstmayr, "A C++ environment for the simulation of multibody dynamics systems and finite elements," in *CD-Proceedings of ECCOMAS Thematic Conference: Multibody Dynamics, Warschau, Poland*, 6 2009.

[7] R. Ludwig and J. Gerstmayr, "Automatic parameter identification for generic robot models," in *Proceedings of the MULTIBODY DYNAMICS 2011 ECCOMAS Thematic Conference*, 7 2011.

[8] H. Gattringer and J. Gerstmayr, Eds., *Multibody System Dynamics, Robotics and Control*. Springer, 2012.

[9] "HOTINT V1.1 - A flexible multibody system dynamics freeware code in C++." [Online]. Available: http://www.hotint.org/

[10] J. Gerstmayr and M. Stangl, "High-order implicit Runge-Kutta methods for discontinuous multibody systems." in *Proceedings of the XXXII Summer School APM2004, June 24-July 1, St. Petersburg, Russia*, 2004, pp. 162–169.

Levels of Integration between Low-Level Reasoning and Task Planning*

Peter Schüller, Volkan Patoglu, Esra Erdem
Faculty of Engineering and Natural Sciences, Sabancı University, İstanbul, Turkey

***Abstract*— We provide a systematic analysis of levels of integration between discrete high-level reasoning and continuous low-level reasoning to address hybrid planning problems in robotics. We identify four distinct strategies for such an integration: (i) low-level checks are done for all possible cases in advance and then this information is used during plan generation, (ii) low-level checks are done exactly when they are needed during the search for a plan, (iii) first all plans are computed and then infeasible ones are filtered, and (iv) by means of replanning, after finding a plan, low-level checks identify whether it is infeasible or not; if it is infeasible, a new plan is computed considering the results of previous low-level checks. We perform experiments on hybrid planning problems in robotic manipulation and legged locomotion domains considering these four methods of integration, as well as some of their combinations. We analyze the usefulness of levels of integration in these domains, both from the point of view of computational efficiency (in time and space) and from the point of view of plan quality relative to its feasibility. We discuss advantages and disadvantages of each strategy in the light of experimental results and provide some guidelines on choosing proper strategies for a given domain.**

I. Introduction

Successful deployment of robotic assistants in our society requires these systems to deal with high complexity and wide variability of their surroundings to perform typical everyday tasks robustly and without sacrificing safety. Consequently, there exists a pressing need to furnish these robotic systems not only with discrete high-level reasoning (e.g., task planning, diagnostic reasoning) and continuous low-level reasoning (e.g., trajectory planning, deadline and stability enforcement) capabilities, but also their tight integration resulting in hybrid planning.

Motivated by the importance of hybrid planning, recently there have been some studies on integrating discrete task planning and continuous motion planning. These studies can be grouped into two, where integration is done at the search level or at the representation level. For instance, [1], [2], [3], [4], [5], [6] take advantage of a forward-search task planner to incrementally build a task plan, while checking its kinematic/geometric feasibility at each step by a motion planner; all these approaches use different methods to utilize the information from the task-level to guide and narrow the search in the configuration space. By this way, the task planner helps focus the search process during motion planning. Each one of these approaches presents a specialized combination of task and motion planning at the search level, and does not consider a general interface between task and motion planning.

*Peter Schüller is supported by TUBITAK 2216 Fellowship.

On the other hand, [7], [8], [9], [10] integrate task and motion planning by considering a general interface between them, using "external predicates/functions", which are predicates/functions that are computed by an external mechanism, e.g., by a C++ program. The idea is to use external predicates/functions in the representation of actions, e.g., for checking the feasibility of a primitive action by a motion planner. So, instead of guiding the task planner at the *search level* by manipulating its search algorithm directly, the motion planner guides the task planner at the *representation level* by means of external predicates/functions. [7], [9] apply this approach in the action description language $\mathcal{C}+$ [11] using the causal reasoner CCALC [12]; [10] applies it in Answer Set Programming (ASP) [13], [14] using the ASP solver CLASP [15]; [8] extends the planning domain description language PDDL [16] to support external predicates/functions (called semantic attachments) and modifies the planner FF [17] accordingly.

In these approaches, integration of task and motion planning is achieved at various levels. For instance, [9], [10] do not delegate all sorts of feasibility checks to external predicates as in [7], [8], but implements only some of the feasibility checks (e.g., checking collisions of robots with each other and with other objects, but not collisions of objects with each other) as external predicates and use these external predicates in action descriptions to guide task planning. For a tighter integration, feasibility of task plans is checked by a dynamic simulator; in case of infeasible plans, the planning problem is modified with respect to the causes of infeasibilities, and the task planner is asked to find another plan.

In this paper, our goal is to better understand how much of integration between high-level reasoning and continuous low-level reasoning is useful, and for what sort of robotic applications. For that, we consider integration at the representation level, since this approach allows a modular integration via an interface, external predicates/functions, which provides some flexibility of embedding continuous low-level reasoning into high-level reasoning at various levels. Such a flexible framework allowing a modular integration is important for a systematic analysis of levels of integration.

We identify four distinct strategies to integrate a set of continuous feasibility checks into high-level reasoning, grouped into two: *directly integrating* low-level checks into high-level reasoning while a feasible plan is being generated, and generating candidate plans and then *post-checking* the feasibility of these candidate solutions with respect to the low-level checks. For direct integration we investigate two

methods of integration: (i) low-level checks are done for all possible cases in advance and then this information is used during plan generation, (ii) low-level checks are done when they are needed during the search for a plan. For post-checking we look at two methods of integration: (iii) all plans are computed and then infeasible ones are filtered, (iv) by means of replanning, after finding a plan, low-level checks identify whether it is infeasible or not; if it is infeasible, a new plan is computed considering the results of previous low-level checks. We consider these four methods of integration, as well as some of their combinations; for instance, some geometric reasoning can be integrated within search as needed, whereas some temporal reasoning is utilized only after a plan is computed in a replanning loop. Considering each method and some of their combinations provide us different levels of integration.

To investigate the usefulness of these levels of integration at representation level, we consider 1) the expressive formalism of HEX programs for describing actions and the efficient HEX solver dlvhex to compute plans, and 2) the expressive formalism of ASP programs for describing actions and the efficient ASP solver CLASP to compute plans. Unlike the formalisms and solvers used in other approaches [7], [8], [9], [10], that study integration at representation level, HEX [18] and dlvhex [19] allow external predicates/functions to take relations (e.g., a fluent describing locations of all objects) as input without having to explicitly enumerate the objects in the domain. Other formalisms and solvers allow external predicates/functions to take a limited number of objects and/or object variables as input only, and thus they do not allow embedding all continuous feasibility checks in the action descriptions. In that sense, the use of HEX programs with dlvhex, along with the ASP programs with CLASP enriches the extent of our experiments.

We perform experiments on planning problems in a robotic manipulation domain (like in [9]) and in a legged locomotion domain (like in [20], [21]). The robotic manipulation domain involve 3D collision checks and inverse kinematics, whereas legged locomotion examples involve stability checks and reachability checks. We analyze the usefulness of levels of integration in these domains, both from the point of view of computational efficiency (in time and space) and from the point of view of plan quality relative to its feasibility.

II. LEVELS OF INTEGRATION

Assume that we have a task planning problem instance H (consisting of an initial state S_0, goal conditions, and action descriptions) in a robotics domain, represented in some logic-based formalism. A history of a plan $\langle A_0, \ldots, A_{n-1} \rangle$ from the given initial state S_0 to a goal state S_n computed for H consists of a sequence of transitions between states: $\langle S_0, A_0, S_1, A_1, \ldots, S_{n-1}, A_{n-1}, S_n \rangle$. A *low-level continuous reasoning module* gets as input, a part of a plan history computed for H and returns whether this part of the plan history is feasible or not with respect to some geometric, dynamic or temporal reasoning.

For example, if the position of a robot at step t is represented as $robot\_at(x, y, t)$ and the robot's action of moving to another location (x', y') at step t is represented as $move\_to(x', y', t)$, then a motion planner could be used to verify feasibility of the movement $\langle robot\_at(x, y, t), move\_to(x', y', t), robot\_at(x', y', t+1) \rangle$. If duration of this action is represented as well, e.g., as $move\_to(x, y, duration, t)$, then the low-level module can find an estimate of the duration of this movement relative to the trajectory computed by a motion planner, and it can determine the feasibility of the movement $\langle robot\_at(x, y, t), move\_to(x', y', t), robot\_at(x', y', t + 1) \rangle$ by comparing this estimate with $duration$.

Let L denote a low-level reasoning module that can be used for the feasibility checks of plans for a planning problem instance H. We consider four different methods of utilizing L for computing feasible plans for H, grouped into two: *directly integrating* reasoning L into H, and *post-checking* candidate solutions of H using L.

For *directly integrating* low-level reasoning into plan generation, we propose the following two levels of integration:

- **PRE** – *Precomputation* We perform all possible feasibility checks of L that can be required by H, in advance. For each failed check, we identify the actions that cause the failure, and then add a constraint to the action descriptions in H ensuring that these actions do not occur in a plan computed for H. We then try to find a plan for the augmented planning problem instance H^{pre}. Clearly, every plan obtained with this method satisfies all low-level checks.
- **INT** – *Interleaved Computation* We do not precompute anything, but we interleave low-level checks with high-level reasoning in the search of a plan: for each action considered during the search, the necessary low-level checks are immediately performed to find out whether including this action will lead to an infeasible plan. An action is included in the plan only if it is feasible. The results of feasibility checks of actions can be stored not to consider infeasible actions repeatedly in the search of a plan. Plans generated by interleaved computation satisfy all low-level checks.

Let us denote by L_{PRE} and L_{INT} the low-level checks directly integrated into plan generation, with respect to PRE and INT, respectively.

Alternatively, we can integrate low-level checks L with H, by means of *post-checking* candidate solutions of H relative to L. We propose the following two methods to perform post-checks on solution candidates:

- **FILT** – *Filtering*: We generate all plan candidates for H. For each low-level check in L, we check feasibility of each plan candidate and discard all infeasible candidates.
- **REPL** – *Replanning*: We generate a plan candidate for H. For each low-level check in L, we check feasibility of the plan candidate. Whenever a low-level check fails, we identify the actions that cause the failure, and then

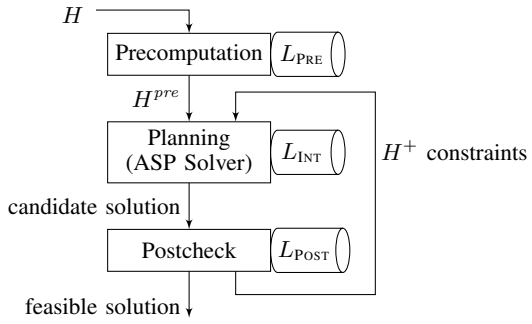


Fig. 1. Components and data flow.

add a constraint to H ensuring that these actions do not occur in a plan computed for H. We generate a plan candidate for the updated planning problem instance H^+ and do the feasibility checks. We continue with generation of plan candidates and low-level checks until we find a feasible plan, or find out that such a feasible plan does not exist.

Let us denote by L_{POST} the low-level checks done after plan generation, with respect to FILT or REPL.

Figure 1 shows the hybrid planning framework we use in this paper to compare different levels of integration, and combinations thereof, on robotics planning scenarios. In particular, Fig. 1 depicts computational components: Precomputation extends the problem instance H using a low-level reasoning module L_{PRE}, Planning integrates a low-level reasoning module L_{INT} into its search for a plan candidate for the problem instance H^{pre} generated by Precomputation. Postcheck uses a low-level module L_{POST} to verify solution candidates (using FILT or REPL) and to potentially add constraints H^+ to the input of Planning.

In our systematic analysis of levels of integration, we do consider this hybrid framework by disabling some of its components. For instance, to analyze the usefulness of PRE, we disable the other integrations (i.e., $L_{\text{INT}} = L_{\text{POST}} = \emptyset$); to analyze the usefulness of a combination of PRE and FILT, we disable other integrations (i.e., $L_{\text{INT}} = \emptyset$).

III. METHODOLOGY

We investigate the usefulness of levels of integration as described above, considering two orthogonal properties: solution quality and planning efficiency. We quantify these properties as follows.

A. Solution Quality

If some low-level module L is not integrated into the planning process, some plan candidates will be infeasible due to failed low-level checks of L. We quantify solution quality by measuring the number of feasible and infeasible plan candidates generated by the search for a plan. This way we obtain a measure that shows how *relevant* a given low-level check is for plan feasibility. Note that with the FILT approach an infeasible plan candidate simply causes a new plan to be generated, while with REPL an infeasible plan candidate causes computation of additional constraints, and a restart of the plan search.

Tightly connected to the number of feasible and infeasible solution candidates is the number of low-level checks that is performed until finding the first feasible plan, and until finding all feasible plans.

B. Planning Efficiency

We quantify planning efficiency by measuring the time required to obtain the first feasible plan, and the time to enumerate all feasible plans. (Note that this includes proving that no further plan exists.)

Independent from the number of low-level checks, the duration of these external computations can dominate the overall planning cost, or it can be negligible. Therefore we measure not only the number of computations of low-level modules but also the time spent in these computations.

IV. DOMAINS AND EXPERIMENTAL SETUP

For our empirical evaluation we use the Robotic Manipulation and the Legged Locomotion domains. Both require hybrid planning. We next give an overview of the domains, their characteristics, and scenarios we used.

A. Robotic Manipulation

We consider a cooperative robotic manipulation problem, as in [9], where two robots arrange elongated objects in a space that contains obstacles. The manipulated objects can only be carried cooperatively by both robots, objects must not collide with each other or the environment, similarly robots must not collide with each other.

A large part of collision checks between objects can already be realized in the high-level representation, however certain checks require usage of geometric models. Collision-freeness between robots for particular collaborative actions can only be determined using low-level geometric reasoning and is not represented in the world model.

Therefore we use two low-level reasoning components to check collision-freeness: given end effector positions of both arms, the L_{rob} module checks whether this combination of positions is collision-free, similarly L_{pay} checks whether an object can be located at a particular pair of coordinates without a collision with the environment. (Additional motion planning to check feasibility of movement from one position to another one can be added as a third check but was omitted in our experiments.) We experiment with 10 instances (over a 11×11 grid) that require plans of upto 20 (average 9.2) steps, and involving up to 58 (average 25.1) actions.

B. Legged Locomotion

In the Legged Locomotion domain, a robot with high degrees of freedom must find a plan for placing its legs and moving its center of mass (CM) in order to move from one location to another one.

For the purpose of studying integration of geometric reasoning with high-level task planning, we created a planning formulation for a four-legged robot that moves on a 10×10

grid. Some grid locations are occupied and must not be used by the robot. Starting from a given initial configuration, the robot must reach a specified goal location where all legs are in contact with the ground.

As legged robots have high degrees of freedom, legged locomotion planning deals with planning in a high-dimensional space. We use a planning problem that is of similar complexity as has been investigated in climbing [21] and walking [22] robots. We also require a feasibility check of leg placement actions. We allow concurrent actions, i.e., moving the center of mass while detaching a leg from the ground, if this does not cause the robot to lose its balance.

We use a low-level reasoning component that determines whether the robot is in a balanced stable equilibrium (L_{bal}), given its leg positions and the position of its CM. We realize this check by computing the support polygon of legs that are currently connected to the ground, and by checking if CM is within that polygon. For these checks we use the `boost::geometry` library to compute a convex hull of all leg positions, and then check whether CM is located within that convex hull.

A second low-level module determines if leg positions are realistic wrt. the position of CM, i.e., if every leg can reach the position where it is supposed to touch to the ground. This check (L_{leg}) is realized as a distance computation between coordinates of legs and CM.

C. Domain Characteristics and Notable Differences

The domains we experiment with exhibit various differences in their characteristics, and such a variety allows us to get practically more relevant results. The most important differences between these two domains are as follows.

Complexity of low-level reasoning. In Legged Locomotion we use a C++ geometric library to perform basic geometric operations which are sufficient for computing check results.

In Robotic Manipulation, object collision checking L_{pay} operates on 3D models of objects and environments additionally L_{rob} requires inverse kinematics to determine the joint configuration of each robot reaching a certain point before performing collision checks between arms.

Hence, in Legged Locomotion, each low-level check requires less time and memory than in Robotic Manipulation. Note that in both domains, we transform discrete 2D grid points to continuous 3D coordinates for low-level reasoning.

Information relevant for low-level reasoning. In Legged Locomotion, we consider problem instances over a 10×10 grid. L_{leg} is a check over two coordinates, therefore there are 10^4 possible L_{leg} checks. The balance check L_{bal} is a totally different situation: we have an input of four leg coordinates and one CM coordinate, therefore, there are 10^{10} possible L_{bal} checks. Such a large number of checks makes precomputation infeasible.

In Robotic Manipulation, both low-level checks are over coordinate pairs on a 11×11 grid; therefore, there are $11^4 \cdot 2 = 29282$ low-level checks.

TABLE I
EFFICIENCY COMPARISON

Integration Method	Overall Time		Low-Level Reasoning	
	FIRST sec	ALL sec	time ALL sec	count ALL #
Robotic Manipulation (10 instances)				
FILT	1716 [2]	1877 [2]	39	724
REPL	2007 [2]	3242 [3]	7	139
PRE	888	974	238	29282
INT	1007	1086	0	467
Legged Locomotion (averages over 20 instances)				
FILT	2434 [5]	3091 [8]	1139	35888
REPL	1345	4192 [9]	12	458
INT	80	133	21	171109
L_{leg}: L_{bal}:				
PRE FILT	2395 [6]	3046 [8]	1272	39354
PRE REPL	1160	4142 [9]	9	324
PRE INT	65	107	23	50677

Numbers in square brackets count timeouts for FIRST resp. ALL.

Based on the number of low-level checks, precomputation for Legged Locomotion seems feasible for only one of the two low-level modules (L_{leg}), while for Robotic Manipulation we can apply precomputation for both low-level computations. Indeed, precomputation for Legged Locomotion can be done in less than 1 second, and for Robotic Manipulation in 238 seconds.[1]

V. EXPERIMENTAL RESULTS

We applied different integration methods to 20 Legged Locomotion and 10 Robotic Manipulation instances of varying size and difficulty.

Tables I and II present results for

- **FIRST**: obtaining the first feasible plan, and for
- **ALL**: obtaining all (maximum 10000) feasible plans.

In our experiments, we use a timeout of 2 hours (7200 seconds) after which we stop computation and take measurements until that moment.

We also limit the number of enumerated plans to 10000 plans. The measurements for enumerating up to 10000 plans reveal information about solution quality and provides a more complete picture of the behavior of each method: one method might find a feasible solutions very fast by chance, whereas finding many or all solutions fast by chance is unlikely.

A. Time Measurements

Table I shows measurements regarding planning efficiency and effort spent in low-level reasoning.

Firstly, it is clear that PRE and INT— the direct integration methods — outperform FILT and REPL— the post-checking methods: for Robotic Manipulation, only PRE and INT are able to enumerate all solutions within the given time limit; for Legged Locomotion, only INT and the PRE/INT combination enumerates all solutions.

[1]All experiments were performed on a Linux server with 32 2.4GHz Intel® E5-2665 CPU cores and 64GB memory.

Comparing the times required by PRE and INT, we see that PRE is more efficient for Robotic Manipulation (888 sec vs 1007 sec on average), which is mainly due to an efficient precomputation technique (see Section V-E).

Even though PRE performs better than INT, it spends more time in low-level reasoning, hence high-level reasoning is faster there; we can explain this by a more constrained search space (low-level check results constraint the search).

After PRE and INT, the next best choice is REPL: it finds solutions to 8 out of 10 instances in Robotic Manipulation, and it finds solutions to all instances for Legged Locomotion, whereas FILT has the same number of timeouts in the Manipulation domain and 5 timeouts for Legged Locomotion. In addition to that, we can see that REPL spends little time in low-level checks compared to other approaches. This is because REPL performs many restarts of the high-level planner which causes it to spend a disproportionate amount of time in high-level planning. Nevertheless, REPL shows its robustness by finding solutions to all but 2 instances.

Finally, FILT fails to find solutions for 7 instances in total which clearly makes it the worst-performing method. The time results for Robotic Manipulation suggest that FILT may be a bit faster than REPL; this may be an effect of some easy instances in that domain where replanning spends more time by reinitialization, than FILT spends by iterating over many similar infeasible solutions. Therefore, even in that domain, we would not suggest to use FILT, as it might — by chance, as low-level reasoning cannot give feedback to high-level reasoning — fail to find a feasible solution for a long time.

B. Effort of Low-Level Reasoning

In Robotic Manipulation, while attempting to enumerate all solutions, FILT performs only 724 low-level checks compared to 29282 checks of PRE. Similarly, in Legged Locomotion, FILT performs 35888 checks and fails to enumerate all solutions for 8 of 20 instances, while INT enumerates all solutions while performing more (171109) low-level checks. Note that these numbers (the last column of Table I) indicate *distinct* low-level reasoning tasks as we cache low-level check results. These numbers show that FILT encounters a small fraction of the low-level checks that are needed to verify all solutions in INT. Caching in fact allows FILT to verify much more actions than INT (numbers not shown), however the number of *distinct* checks (numbers shown) is higher in INT. We conclude that INT traverses the solution space much more efficiently.

In Legged Locomotion, low-level checks depend on a large part of the candidate plan, so caching is not as effective as in Robotic Manipulation. This, together with the fact that in FILT the high-level is not guided by low-level checks, causes the FILT approach to spend more time in low-level reasoning than other approaches.

Note that, to obtain a reasonable comparison between PRE and the other approaches, we include times and counts of precomputed low-level checks in Table I (which explains the large values for low-level computations in these rows).

TABLE II
SOLUTION QUALITY COMPARISON

Integration Method	Infeasible Candidates FIRST #	Infeasible Candidates ALL #	Plans found ALL #	Feasible Plans ALL %
Robotic Manipulation (averages over 10 instances)				
FILT	586	11787	622	<0.1
REPL	11	38	621	94.2
PRE	0	0	652	100.0
INT	0	0	652	100.0
Legged Locomotion (averages over 20 instances)				
FILT	11282	35487	360	1.0
REPL	28	68	250	78.6
INT	0	0	1394	100.0
L_{leg}: L_{bal}:				
PRE FILT	10938	39116	340	0.9
PRE REPL	31	69	255	78.7
PRE INT	0	0	1394	100.0

C. Solution Quality

Methods PRE and INT do not generate infeasible solution candidates, as they use all low-level checks already in search.

If we compare the number of infeasible solution candidates of FILT and REPL in Robotic Manipulation, we observe that FILT generates mainly infeasible solution candidates compared to the number of feasible solutions (11787 vs 622) while REPL creates only 38 infeasible candidates while enumerating 621 feasible plans.

In Legged Locomotion, the results for FILT are similar, however REPL performs a bit worse than in Robotic Manipulation with 250 infeasible candidates compared to 68 feasible solutions. A possible reason for this difference could be the same reason why PRE is not feasible in that domain: there is a large amount of possible inputs to L_{bal} compared to the other low-level checks we used. Due to the large input space, each failed L_{bal} check constrains the search space only by a small amount, so REPL produces more infeasible solutions than in Robotic Manipulation.

D. Memory Usage

We measured peak memory usage over the whole runtime of each instance. Interleaved computation with the dlvhex solver (columns with INT) requires an average of around 2000MB, the maximum stays below 4000MB. For non-interleaved computations, GRINGO and CLASP were connected with low-level checks using Python scripts. These approaches require around 400MB of memory with a maximum below 1000MB.

E. Combination of PRE with other methods

As shown in the Legged Locomotion experiments, PRE can be combined with other approaches. In our experiments we observe that adding PRE increases efficiency.

However, PRE adds a fixed cost to solving because it needs to precompute many points. Depending on efficiency of low-level computations, even if there are few possible

input combinations to low-level modules, precomputation might be infeasible.

It can be feasible to precompute a large set of related low-level checks more efficiently than performing a check for each possible input combination. In our experiments we implemented such a *dedicated precomputation* method: for Robotic Manipulation it requires 238 seconds in total, while calling individual checks requires 1361 seconds in total. Hence without dedicated efficient precomputation, PRE performs much worse than INT.

VI. Discussion and Conclusion

Our experiments suggest the following conclusions. If robust and highly complex reasoning is required, and if this reasoning is done frequently (so that performance gains will become relevant) then using full interleaved reasoning (INT) is the only good option. INT has the best performance with respect to run times, and it can enumerate most solutions compared to other approaches. The reason is that INT uses only those low-level checks which are necessary (they are computed on demand) and therefore does not overload the solver with redundant information (as PRE does). Furthermore, INT considers failed checks in the search process and thereby never picks an action where it is known that the action will violate a low-level check. This is similar as in the REPL approach, but much more efficient as the integration is much tighter compared to REPL. However, the performance of INT comes at a price: (a) it requires more memory than generating solution candidates and checking them afterwards, and (b) it requires a solver that allows for interacting with the search process in a tight way, usually through an API that has to be used in a sophisticated way if it shall be efficient.

If reasoning operates on a manageable amount of inputs, such that precomputation is a feasible option, then PRE is a good choice. In our Robotic Manipulation experiments, PRE outperforms all other methods, which is partially due to our using a dedicated efficient precomputation tool (see Section V-E). In Legged Locomotion, combining PRE with other methods also increased efficiency.

The FILT approach performs the worst, because nothing guides the search into the direction of a feasible solution; FILT is not robust and enumerates many infeasible solutions.

If both PRE and INT are not possible[2] then REPL should be used; this approach does not have the same performance as INT and PRE, however it is a very robust approach as it is guided by its wrong choices — we can think of the constraints that are added for failed low-level checks as the approach 'learning from its mistakes'. The benchmark results for Legged Locomotion clearly show the robustness of REPL compared to FILT: the former finds solutions for all problems, the latter only for 15 out of 20 instances.

A possible improvement to REPL could be to let it enumerate a certain amount of solutions to gather more constraints, then add all these constraints and restart the search. This is a hybrid approach between FILT and REPL. Selecting the right moment to abort enumeration and restart the solver is crucial to the performance of such a hybrid approach, and we consider this a worthwhile subject for future investigations.

[2]This can be the case if the state of inputs is large, and additionally we need to use a certain planner software that does not support interleaved computation.

References

[1] F. Gravot, S. Cambon, and R. Alami, *Robotics Research The Eleventh International Symposium*, ser. Springer Tracts in Advanced Robotics. Springer, 2005, vol. 15, ch. aSyMov:A Planner That Deals with Intricate Symbolic and Geometric Problems, pp. 100–110.

[2] K. Hauser and J.-C. Latombe, "Integrating task and PRM motion planning: Dealing with many infeasible motion planning queries," in *Workshop on Bridging the Gap between Task and Motion Planning at ICAPS*, 2009.

[3] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Proc. of ICRA*, 2011, pp. 1470–1477.

[4] E. Plaku and G. D. Hager, "Sampling-based motion and symbolic action planning with geometric and differential constraints," in *Proc. of ICRA*, 2010, pp. 5002–5008.

[5] J. Wolfe, B. Marthi, and S. Russell, "Combined task and motion planning for mobile manipulation," in *International Conference on Automated Planning and Scheduling*, 2010.

[6] E. Plaku, "Planning in discrete and continuous spaces: From LTL tasks to robot motions," in *Proc. of TAROS*, 2012, pp. 331–342.

[7] O. Caldiran, K. Haspalamutgil, A. Ok, C. Palaz, E. Erdem, and V. Patoglu, "Bridging the gap between high-level reasoning and low-level control," in *Proc. of LPNMR*, 2009.

[8] A. Hertle, C. Dornhege, T. Keller, and B. Nebel, "Planning with semantic attachments: An object-oriented view," in *Proc. of ECAI*, 2012, pp. 402–407.

[9] E. Erdem, K. Haspalamutgil, C. Palaz, V. Patoglu, and T. Uras, "Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation," in *International Conference on Robotics and Automation*, 2011, pp. 4575–4581.

[10] E. Erdem, E. Aker, and V. Patoglu, "Answer set programming for collaborative housekeeping robotics: representation, reasoning, and execution," *Intelligent Service Robotics*, vol. 5, pp. 275–291, 2012.

[11] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner, "Nonmonotonic causal theories," *AIJ*, vol. 153, p. 2004, 2004.

[12] N. McCain and H. Turner, "Causal theories of action and change," in *Proc. of AAAI/IAAI*, 1997, pp. 460–465.

[13] V. Lifschitz, "What is answer set programming?" in *Proc. of AAAI*. MIT Press, 2008, pp. 1594–1597.

[14] G. Brewka, T. Eiter, and M. Truszczynski, "Answer set programming at a glance," *Commun. ACM*, vol. 54, no. 12, pp. 92–103, 2011.

[15] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub, "clasp: A conflict-driven answer set solver," in *Proc. of LPNMR*, 2007, pp. 260–265.

[16] M. Fox and D. Long, "Pddl2.1: An extension to pddl for expressing temporal planning domains," *J. Artif. Intell. Res. (JAIR)*, vol. 20, pp. 61–124, 2003.

[17] J. Hoffmann and B. Nebel, "The ff planning system: Fast plan generation through heuristic search," *J. Artif. Intell. Res.*, vol. 14, pp. 253–302, 2001.

[18] T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits, "A Uniform Integration of Higher-Order Reasoning and External Evaluations in Answer-Set Programming," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2005, pp. 90–96.

[19] T. Eiter, G.Ianni, R.Schindlauer, and H.Tompits, "Effective integration of declarative rules with external evaluations for Semantic-Web reasoning," in *Proc. of ESWC*, 2006.

[20] T. Bretl, S. Lall, J.-C. Latombe, and S. M. Rock, "Multi-step motion planning for free-climbing robots," in *Algorithmic Foundations of Robotics VI*, ser. Springer Tracts in Advanced Robotics, vol. 17. Springer, 2005, pp. 59–74.

[21] T. Bretl, S. M. Rock, J.-C. Latombe, B. Kennedy, and H. Aghazarian, "Free-climbing with a multi-use robot," in *ISER*, ser. Springer Tracts in Advanced Robotics, M. H. A. Jr. and O. Khatib, Eds., vol. 21. Springer, 2004, pp. 449–458.

[22] K. K. Hauser, T. Bretl, J.-C. Latombe, K. Harada, and B. Wilcox, "Motion planning for legged robots on varied terrain," *I. J. Robotic Res*, vol. 27, no. 11-12, pp. 1325–1349, 2008.

From tendrils to robots: kinematic study for a bio-inspired grasping system

Renato Vidoni[1], Tanja Mimmo[1] and Camilla Pandolfi[2]

***Abstract*— In this work, the robotic grasping problem is addressed with a bio-mimetic approach.**

Looking at the capabilities of some special climbing plants, the tendril behavior has been focused, modeled and simulated from a kinematic point of view.

First of all, the three main tendril movements, i.e. circumnutation, coiling and free-coiling, have been evaluated; after that, a modular kinematic model able to take into account the main tendril features has been defined; then, the kinematics of the system has been solved and a tendril-kinematic simulator implemented.

I. INTRODUCTION

In robotics, the grasping ability and the design and control of effective systems such as mechanical hands have been studied since the eighties [18], [17].

Tactile sensing, restraining (fixturing) and manipulating with fingers (dexterous manipulation) are usually addressed and studied. Robotic grasping systems often obtain relevant objects information through vision systems and/or assume as known most or all the information needed for the grasping [19], [4]. Robot mechanical hands are usually simple, without anthropomorphic intent (i.e. grippers, jaws, compliance devices), and developed to perform specific tasks (i.e. suction cups for keeping glass, compressed air-driven tongs). Broad reviews on these topics can be found in [1], [20], [24]. From the mechatronic point of view, a lot of work has been made to create, optimize and miniaturize tactile sensors solutions [24]. If nature is considered as a source of inspiration, many systems that show a sort of reflex-like behavior for mediating with the objects to grasp can be found. Thus, from the bio-mimetic point of view, in particular for coiling robotics systems, some work has been done to replicate the capabilities of some special animals, i.e. robotic snakes. The Hirose's group developed a class of rigid-link hyper redundant dexterous manipulators called "active cord mechanism" (ACM) or serpentine robots; the "Continuum manipulators" [16] show a backbone structure system with a high number of joints and a very short length of links to tend to an ideal continuum condition that can bend and contract/extend in any point (e.g. [2], [11]). Tendons and artificial muscle technologies are among the most effective hardware realizations. Successful works on backbone manipulators are also the octopus and the arms and tentacles of squid - inspired robots [22], e.g. the OCTARM robot and the tentacles-inspired robot [12], [23], e.g. AirOctor. Recently, the Octopus project (EU-FP7) has studied novel technologies for high dexterity robots inspired by the octopus [25]. Indeed, the octopus tentacles are able to carry out an effective grasping by contracting and elongating muscles and fibers and bending in the desired direction [3], [14], [9], [13].

In nature other systems show effective grasping capabilities. Indeed, some climbing plants, at today not yet well studied both from a physiological and bio-mimetic purposes, are able to find, recognize and grasp supports by means of specialized filiform organs called tendrils. This work aims at focusing the tendril behavior for a bio-mimetic purpose in particular by studying the main plant behaviors from a kinematic perspective.

*This work was supported by the ESA-Advanced Concepts Team (ARIADNA study ID 12/6402).

[1]R. Vidoni and T. Mimmo are with the Faculty of Science and Technology, Free University of Bozen-Bolzano, Bolzano, Italy renato.vidoni, tanja.mimmo at unibz.it

[2] C. Pandolfi is with the ESA-Advanced Concepts Team, European Space Research and Technology Centre, Noordwijk, The Netherlands camilla.pandolfi@unifi.it

II. TENDRILS

Tendrils are long, slender, filiform, irritable organs, derived from stems, leaves, or flower peduncles [5] which may occur either as un-branched or multi-branched organs with a variable length.

Tendrils usually show three main movements [10]:

- circumnutation, an endogenous movement increasing the probability of contact with supports,
- contact coiling, in which the stimulated tendril coils around a support, and
- free-coiling, in which the tendril develops helical coils along its axis, not necessarily as a result of stimulation.

A. Circumnutation

Circumnutation is an oscillating growth pattern in rapidly elongating plant organs, such as roots, shoots, branches and flower stalks. Circumnutational oscillations are manifestations of the radially asymmetric growth rate, typical of elongating plant organs [7], [15] . Although circumnutatory movements appear to have no useful purpose in the majority of the cases, in climbing plants, they have a crucial function in seeking mechanical supports. In fact, the circumnutational movement sweeps the tendril in the space, e.g. circular path, increasing the possibility to contact with a support.

B. Coiling and Grasping

Contact coiling or simply coiling initiates as a response to a local mechanical stimulus of the tendrils which start curling around a support and tightening up [21].

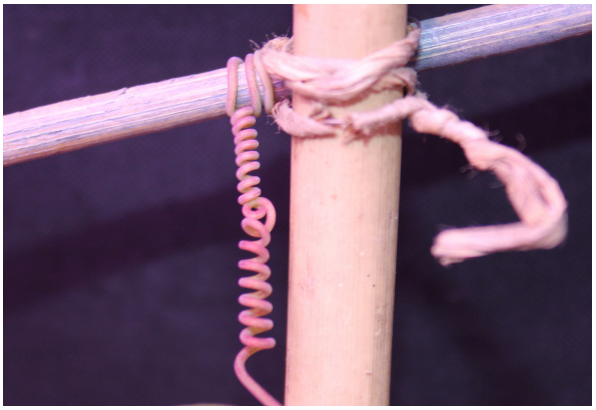


Fig. 1. Coiling and free-Coiling

Since tendrils are often modified leaves or stems, the contact coiling allows the plant to gain height or to turn the leaves maximizing sun exposure for the photosynthetic activity. If no suitable contact or support is found, the tendril might even uncoil indicating that the process is reversible. The tendril seems to perceive the stimulus/touch locally in the epidermal cells activating chemical signaling within the whole plant organ; the response is very fast leading to a coiling within seconds.

C. Free-coiling

Once the tendril has grasped around an object or support, the plant organ undergoes a secondary coiling, free-coiling, which brings the plant closer to the support by creating an elastic spring-like connection. This spiral structure has very often been compared to a telephone cord and might be described by an ideal helical spring. Darwin [6] observed that there are the same numbers of spirals in both directions in order to compensate the twisting of the axis. If there is no grasp, the tendril curves and creates a sort of spiral.

Fig.1 shows a tendril after the coiling and free-coiling phases.

III. Key features

Concerning the grasping and free-coiling phases of the plant tendrils, to our knowledge, no bio-mimetic results or attempts can be found in literature.

Indeed, if the grasping and free-coiling phases are considered, important differences or not yet discovered similarities between the plant and animal behavior can be appreciated:

- Tendrils motion and contact recognition
  Tendrils, in particular Passiflora tendrils, are able to recognize supports and obstacles on the overall surface thanks to the presence of specialized fibers. Indeed, tendrils bend in different directions and not only on one side, especially in the second half of the length.
- Multiple coils
  This capability can be viewed looking at a winch-capstan system. Multiple coils allow to have a small and negligible tension in the tendril apex, to have a tension that increases from the apex to the most inner touching point and to avoid the slippage due to the increased touching surface and the related friction.
- Reflex - modular distributed behavior
  The most sensitive part of the tendril is its second half. When it touches or is touched in this zone, it shows a reflex behavior, i.e. it bends. The signal transmission shows a sort of modular behavior. This means that the zone that senses a support induces a contraction phase to the near fibers; after that, if the contact increases, i.e. the touched nearest zones sense a contact, the bending signal is transmitted creating the overall tendril motion and grasping. Thanks to this, a distributed reflex control is made, allowing the activation of the motion locally and only when necessary.
- Object searching
  The circumnutation phase, in which all the tendril moves, is a manifestation of the radially asymmetric growth rate and results in a sweep "seeking" movement that can, for our bio-mimetic purposes, be simplified in a movement created by a proper active joint at the tendrils base.
- Spring actuator
  The free-coiling phase allows to pull the stem towards the grasped support. By coming back to the original/intrinsic helical shape, the tendril shortens the distance between the fixed end-points. Moreover, the helical-spring shape is perfectly tuned to resist to external loads and disturbances.

These features, together with the strengths related to the adoption of flexible manipulator systems, strongly support the study and design of a plant-inspired robotic tendril.

IV. Kinematic modeling

To design a bio-robotic tendril, the overall structure has been considered from a kinematic point of view. The model has been conceptualized and simplified dividing the tendril in two main parts taking into account the different stimulus sensitivity along the tendril length: the first, Free-Coiling (FC), mainly devoted to the free-coiling and pulling phase, and the second, Grasping-Coiling (GC), devoted to the coiling and grasping phase (Fig. 2).

Thanks to this, the GC part can be subdivided in independent modules able to react when in touch or hit by something, and bend; the FC part, devoted to the pulling, can be viewed as a single actuator (eventually made of multiple sections) that changes its shape from a linear wire to a helical spring.

To model the GC part, the approach followed by Jones and Walker [11] and Hannah and Walker [8] has been chosen. They worked on the kinematics of multi-section continuous robots. In this way, at each sub-section can be associated a kinematic description by means of elementary pairs/joints, the Denavit-Hartenberg (DH) notation can be exploited, and a reflex modular motion simulated. Moreover, a direct relation between a "DH" robotic section and a wire driven module can be defined allowing to compute the cable tensions for a particular sub-section motion.

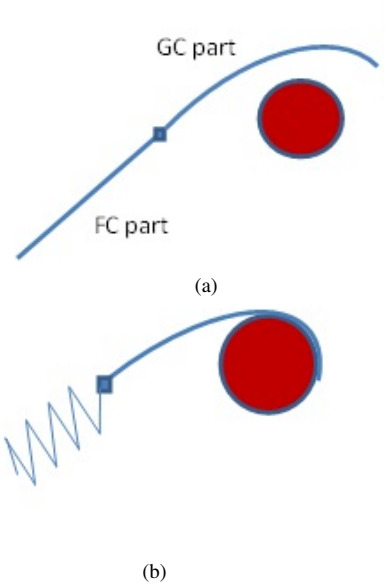


Fig. 2. GC and FC modules of the bio-mimetic tendril model

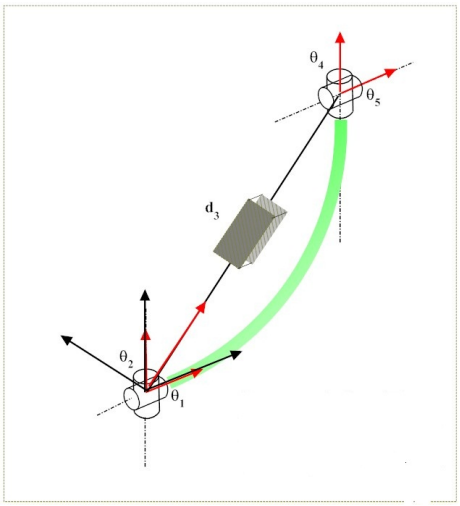


Fig. 3. Tendril kinematic section

The idea is to fully model the kinematics of the tendril GC part using closed-form equations. This is carried out by dividing it in sections/modules which can at least bend in two dimensions.

A. Grasping Part

The GC part has been considered as subdivided in n-sections. The main assumption is that the tendril section has a constant curvature. A tendril-based continuous system lacks joints and each tendril section can be modeled through an arc of constant curvature, i.e. by means of parameters such as length, curvature and/or angle of curvature. To shift between this formulation and a conventional DH formulation, an equivalent rigid-link-joint section and the transformation relations have to be defined; thus, to fit a conventional rigid-link manipulator to the tendril section, the relationship $[\theta,\boldsymbol{d}]^T = f(l,\kappa,\varphi)$, where θ and $\boldsymbol{d}$ are the DH parameters, l the length section, κ the curvature and φ the angle of curvature, has to be found.

For each section, a simplified robotic system made of rigid links and joints is defined. Fig. 3 shows the "equivalence" between the real wire/tendril section and the robotic-tendril.

The section is modeled with a first universal joint, i.e. two revolute pairs with orthogonal and intersecting axis of rotation, a prismatic joint, and a second universal joint; this last joint and the related variables are coupled with those of the first joint of the following section. The first joint allows to rotate the local frame axis to be oriented to the section tip, the prismatic joint allows the translation to the last point of the section, i.e. the final coordinate frame origin, and the last pair allows to correctly rotate and orient the local frame with the following section. Since the last pair of a section and the first pair of the following are coupled, the independent variables result in three per section. If the number of sections is sufficiently high, the prismatic joint can be constrained, since the error made becomes negligible. A fixed rotation on the x-axis at the beginning of the DH table is added to have the section extension along the z-axis; moreover, a fixed rotation at the end of the DH table allows to correctly orient the tip. The equivalent robotic-tendril independent variables are chosen in order to properly simulate the real plant behavior. Tab.I shows the DH parameters related to a tendril section shown in Fig. 3.

TABLE I
DH TABLE FOR A TENDRIL SECTION

Link	a	α	d	θ
-	0	$\pi/2$	0	0
1	0	$\pi/2$	0	θ_1
2	0	$\pi/2$	0	$\theta_2 + \pi/2$
3	0	$-\pi/2$	d_3	0
4	0	$-\pi/2$	0	$\theta_4 + \pi/2$
5	0	0	0	θ_5
-	0	$-\pi/2$	0	0

The resulting homogeneous transformation matrix for each section of the GC part is:

$$\mathbf{A} = \begin{bmatrix} -c_1 s_2 s_4 c_5 + c_1 c_2 c_4 c_5 - s_1 s_5 & c_1 s_2 s_4 c_5 + c_1 c_2 c_4 s_5 - s_1 c_5 & -c_2 s_4 c_1 + s_2 c_4 & c_1 c_2 d_3 \\ -s_1 s_2 s_4 c_5 + s_1 c_2 c_4 c_5 - c_1 s_5 & s_1 s_2 s_4 s_5 + s_1 c_2 c_4 s_5 - c_1 c_5 & -s_1 c_2 s_4 + s_2 c_4 & s_1 c_2 d_3 \\ (c_2 s_4 + s_2 c_4) s_5 & -(c_2 s_4 + s_2 c_4) s_5 & c_2 c_4 + s_2 s_4 & s_2 d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $c_i = \cos(\theta_i)$ and $s_i = \sin(\theta_i)$.

By simplifying the system and supposing a planar 2-D motion, the curve becomes planar with a constant curvature. It can be associated to a planar robotic system made of two revolute joints with axis of rotation perpendicular to the plane, linked with a prismatic joint. In such a case, the DH table results as in Tab. II.

The model is thus highly flexible. This means that it can both be exploited to simulate the real tendril behavior by

increasing the number of sections, and a modular robotic system made of independent reflexive sections. In the latter case, the robotic tendril is seen as a chain of independent embedded modules that can be realized by means of motorized actuators or wire-spring driven systems. Hence, the kinematic model can represent the base for the evaluation and control of the real system prototype.

TABLE II
DH TABLE FOR A TENDRIL PLANAR SECTION

Link	a	α	d	θ
1	0	-π/2	0	θ_1
2	0	π/2	d_3	0
3	0	0	0	θ_5

B. Free-Coiling Part

The FC part can be adequately modeled by a unique kinematic section. Indeed, on one hand it is constrained by the stem/fixed base and, on the other hand, to the contact/grasping point.

The chosen section has the same characteristics of the previous defined ones, both for the 3D and 2D motion. Indeed, thanks to the first universal joint the tendril can be properly oriented; the prismatic joint allows to consider the free-coiling main effect, and the second universal joint to take into account the correct final orientation for connecting the two bio-mimetic tendril parts. Since the two extreme points are constrained and no torsion has to be allowed for both the base and the connection with the GC part, the two universal joints are passive joints; this means that the related angular values are defined once ended the grasping phase. The unique active joint is the prismatic pair; its length value is the distance between the base and the origin of the first local coordinate frame of the GC section.

From the kinematic point of view, the problem consists in a shift of point of view: the end-effector of the section becomes the base, the stem, and the movement has to be described with respect to the local reference frame of the last pair of the section, i.e. the one that now is constrained and fixed.

V. KINEMATIC SIMULATOR

The derived formulas have been implemented in a Matlab simulator to simulate the kinematics of a bio-inspired tendril.

The FC section has been defined with a greater length with respect to the GC sections, e.g. half of the overall length, and considered in the searching and grasping phase constrained in its origin point. The GC section has been divided in *n* sections of equal length. The circumnutation phase has been simplified in a centralized motion: only the first universal joint of the chain is driven; this means that the overall tendril spans a cone in a 3D motion (or an angle in 2D) searching a support to be grasped.

The grasping phase has been conceived as a reflex behavior considering, for this purpose, each GC section as an independent module. The prismatic joint has been considered as fixed in length since a large number of sections are simulated. This means that, if a GC section touches an obstacle, it stops the main circumnutation motion; after that, it reacts by actuating the last joint of the section in the direction of the stimulus. This results in bending the remaining tendril in the stimulus direction until when either another section recognizes a stimulus, i.e. is in touch with the support, or the minimum curvature angle is reached. In the former case the bending motion continues and the grasping by coiling goes on; in the latter case the motion stops and the tendril motion has to be zeroed/restarted (Fig.s4,4(f)).

The contact has been implemented by searching for each module if there is intersection between the segment that connects the two universal/revolute joints, i.e. d_3, and the surface to be find, e.g. a cylinder (Fig. 4). Two particular conditions have been implemented: the minimum and maximum radius. The former is related to the intrinsic curvature of the tendril that represents the minimum radius of the object that can be coiled, the latter is related to the slippage limit, i.e. the limit over which the tendril slips on the surface and curls. The DH independent variables are chosen to simulate a smooth and slow motion, i.e. a small angular step.

The FC behavior has been implemented as a pulling motion driven by the prismatic joint. In such a case, the base of the section becomes the connection point between the GC and FC parts, while the end-effector becomes the coordinate frame of the stem (Fig.5).

The model can then serve for both replicating the tendril motion with the aim to better understand its behaviors, and designing a robotic system made of independent reflexive sections. These can be conceptualized and realized in a standard way by means of motorized actuators and, with a light and simple system purpose, by means of wires and/or smart materials. In particular, smart material such as Smart Memory Alloys (SMA) could serve as the technological base for developing independent modules while miniaturized touching and force sensors can be exploited for the contact recognition phase.

VI. CONCLUSIONS

In this work, the tendril capabilities of finding and coiling around a support, together with the ability to pull the stem towards the grasped object have been evaluated in a bio-robotic perspective. The three main tendril behaviors have been investigated, simplified and modeled from a kinematic perspective by dividing the system in two main sections: one devoted to the grasping and the other to the free-coiling (pulling). A robotic formulation has been defined and the kinematics, by means of the DH formulation, solved and simulated. With this bio-inspired model, a first evaluation of the tendril modular reflex behavior can be done and possible realization ideas of a tendril-robot can be evaluated.

REFERENCES

[1] A Bicchi and V Kumar, *Robotic grasping and contact: a review*, Proceedings of IEEE International Conference on Robotics and Automation, 2000, ICRA '00, 1, 348 - 353, San Francisco, CA , USA , 2000

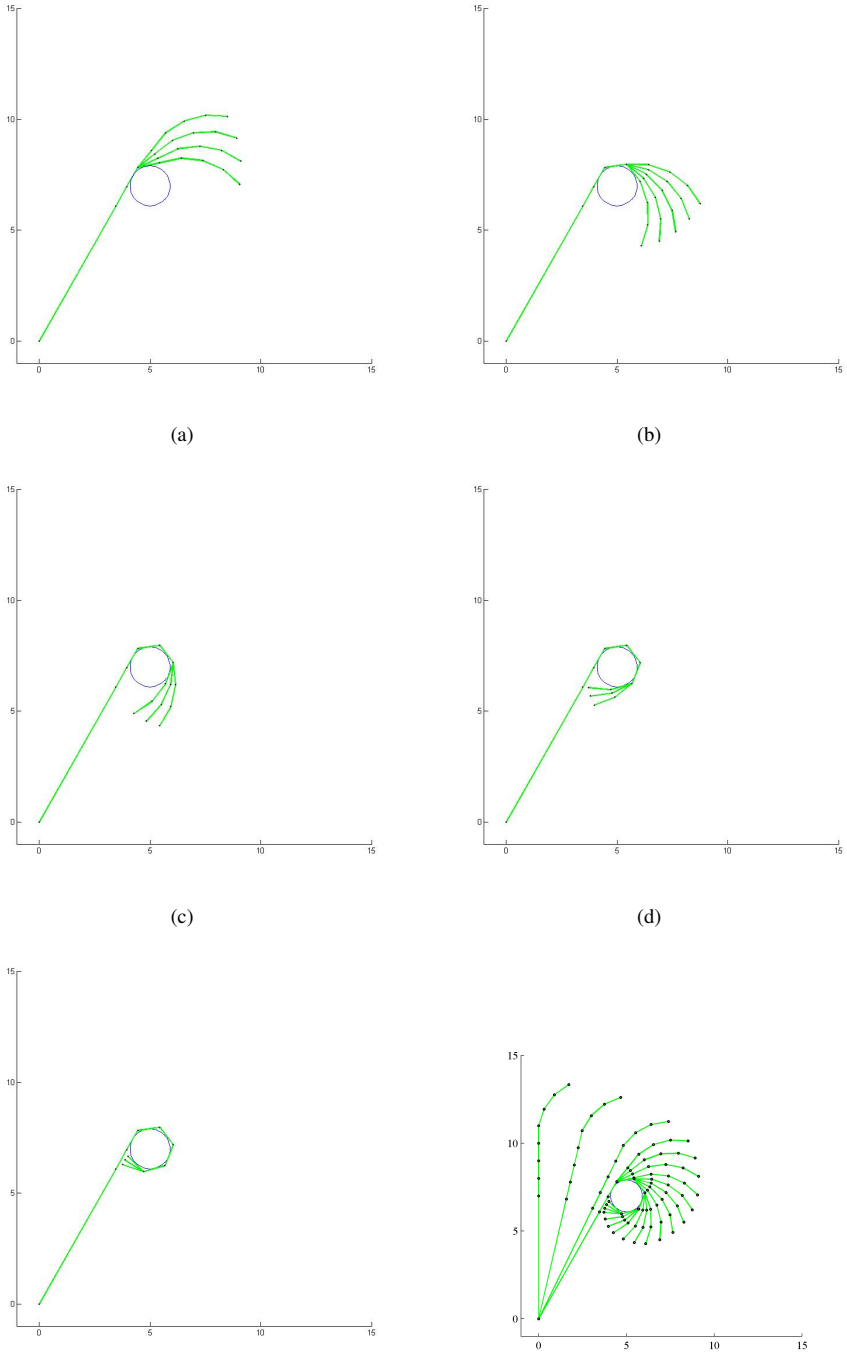
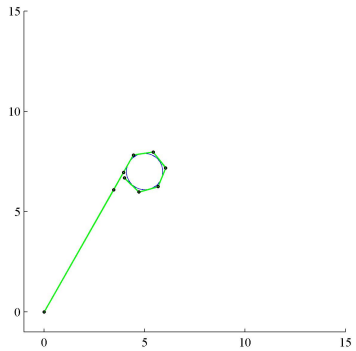
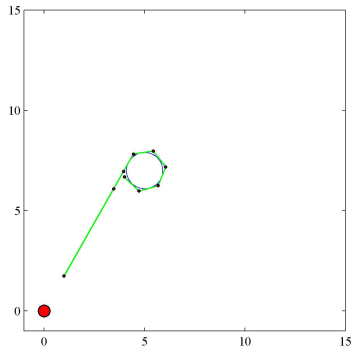


Fig. 4. Bio-mimetic tendril kinematics: grasping



(a)



(b)

Fig. 5. Bio-mimetic tendril kinematics: free-coiling

[2] R Buckingham, *Snake arm robots*, Ind. Robot Int. J. 29(3), 242-245, 2002.

[3] M Calisti, M Giorelli, G Levy, B Mazzolai, B Hochner, C Laschi and P Dario, *An octopus-bioinspired solution to movement and manipulation for soft robots*, Bioinsp. Biomim. 6, 036002, 2011

[4] G Cannata and M Maggiali, *An embedded tactile and force sensor for robotic manipulation and grasping*, Humanoid Robots, 5th IEEE-RAS International Conference, 2005.

[5] C Darwin, *On The Movements and Habits of Climbing Plants*, London: John Murray, 1865.

[6] C Darwin, *The movements and habits of climbing plants*, Appleton, New York, New York, 1876.

[7] C Darwin and F Darwin, *The power of movements in plants*, Appleton, New York, New York, 1880.

[8] MW Hannan and ID Walker, *Kinematics and the Implementation of an Elephant's Trunk Manipulator and other Continuum Style Robots*, Journal of Robotic Systems, 20: 45-63, 2003

[9] JP Hou, RHC Bonser and G Jeronimidis, *Design of a Biomimetic skin for an octopus- inspired robot- Part I: Characterising octopus skin*, Journal of Bionic Engineering, 8: 288-296, 2011

[10] MJ Jaffe and AW Galston, *The physiology of tendrils*, Annu. Rev. Plant. Physiol. 19:417-434, 1968.

[11] BA Jones, ID Walker, *Kinematics for Multisection Continuum Robots*, IEEE Transaction on Robotics, 22:1, Feb 2006

[12] BA Jones, ID Walker, *Practical Kinematics for Real-Time Implementation of Continuum Robots*, IEEE Transaction on Robotics, 22:6, Dec 2006

[13] R Kang, DT Branson, E Guglielmino and DG Caldwell, *Dynamic Model and Control of a Multiple Continuum Arm Robot Inspired by Octopus*, In Computers and Mathematics with Applications, 2012

[14] C Laschi, B Mazzolai, M Cianchetti, L Margheri, M Follador and P Dario, *A Soft Robot Arm Inspired by the Octopus*, Adv. Robotics 26 709-726, 2011

[15] S Mugnai, E Azzarello, E Masi, C Pandolfi, S Mancuso, *Nutation in plants*. In: Mancuso S, Shabala S, editors. Rhythms in plants: phenomenology, mechanisms and adaptative significance. Springer; 2007. pp. 7790.

[16] G Robinson and JBC Davies, *Continuum robots - a state of the art*, IEEE Int. Conf. Robot. Automation, 2849-2854, Detroit (USA), 1999

[17] JK Salisbury, *Kinematic and Force Analysis of Articulated Hands*. Ph.D. Thesis (Stanford University, Stanford 1982)

[18] B Siciliano, O Khatib, *Handbook of Robotics*, Springer, 2008.

[19] T Takahashi T, T Tsuboi, T Kishida, Y Kawanami, S Shimizu, M Iribe, T Fukushima and M Fujita, *Adaptive Grasping by Multi Fingered Hand with Tactile Sensor Based on Robust Force and Position Control*, IEEE International Conference on Robotics and Automation Pasadena,CA, USA, May 19-23, 2008

[20] J Tegin and J Wikander, *Tactile sensing in intelligent robotic manipulation - a review*, Industrial Robot: An International Journal, Volume 32, Number 1, 2005 , pp. 64-70(7)

[21] KC Vaughn and AJ Bowling, *Biology and Physiology of Vines*, Horticultural Reviews, Volume 38, 2011.

[22] I Walker, D Dawson, T Flash, F Grasso, R Hanlon, B Hochner, W Kier, M Pagano, C Rahn and Q Zhang, *Continuum robot arms inspired by cephalopods*, Proc. SPIE, Vol. 5804 (2005) pp. 303-314

[23] I Walker, C Carreras, R McDonnell and G Grimes, *Extension versus bending for continuum robots*, Int. J. Adv. Robot. Syst. 3(2), 171-178, 2006

[24] H Yousef, M Boukallel and K Althoefer, *Review-Tactile sensing for dexterous in-hand manipulation in robotics - A review*, Sensors and Actuators A 167 (2011) 171-187

[25] EU Octopus FP7 project, *http://www.octopus-project.eu/* accessed on December 2012

Ambient Assitive Technologies: The mobile robot P3AAT

Richard Wagner, Peter Wolff, Klaus Schäffer and Friedrich Praus
University of Applied Sciences Technikum Wien, Department of Embedded Systems
Höchstädtplatz 6, 1220 Vienna
{wagnerr, es11m010, es11m009, friedrich.praus}@technikum-wien.at

***Abstract*— This paper reports preliminary results on a student project dealing with the development of an autonomous mobile service robot. The robot is able to navigate autonomously, detect bottles in different shapes and grasp them. The paper describe the hardware components used by the robot and the software architecture based on the Robot Operation System.**

I. INTRODUCTION AND MOTIVATION

The demographic trend in industrial nations towards an aged population and a decline in birth rates requires new strategies to cope with the upcoming social and economic challenges. Mobile service robotics can provide support to elderly or disabled persons, foster their autonomy and quality of life and lower the cost of medical care.
This paper describes the hardware and software architecture of a mobile service robot called P3AAT, shown in Figure 1. The robot is named after the the Robot Base Platform P3-AT and Ambient Assistive Technologies (AAT). The P3AAT should be able to support elderly people in a daily life routine to detect a bottle on a table, grasp it and bring it back to them. Anyway, this mobile platform should be the starting point for further student projects dealing with navigation, object detection or object grasping. The P3AAT is modular in design and function, so it can quickly and easily be customized to possible future projects.
This paper is structured as follows: Section 2 presents the hardware configuration and Section 3 describes the software architecture of the robot P3AAT. Section 4 evaluates the results and the experiences gained during the software integration and testing phase. Finally Section 5 concludes the paper.

II. HARDWARE

This Section describes the hardware-architecture of the robot P3AAT. As Figure 2, shows the architecture consists of two separate PCs: A Main-PC on the mobile service robot and an Remote-PC to access the robot. A joystick is connected via USB with the remote PC for manual control of the robot. The depth sensors Microsoft Kinect and Asus Xtion, the laser scanner and the robotic arm are connected to the Main-PC on the mobile robot via USB or Ethernet. The two PCs are connected to each other via a WLAN.

*The work presented in this paper was funded by the City of Vienna, department MA23, under grant number MA23-Projekt 10-04.
**The authors wish to thank the Department of Computer Science for providing the hardware used in this project.

Fig. 1. The Robot P3AAT at the attempt to grasp the bootle.

The base platform Pioneer P3AT, which is described in detail in the next Chapter, contains a microcontroller. The sonar sensors and the bumpers are directly connected to this micro-controller. In the beginning of the project a tilting Hokuyo URG-04LX laser scanner was used to perceive the environment. The laser scanner was replaced by two depth sensor because of higher costs. In the current configuration the robot P3AAT is equipped with three depth sensors: Two depth sensors are used for navigation purposes and one depth sensor is used to detect the bottle.

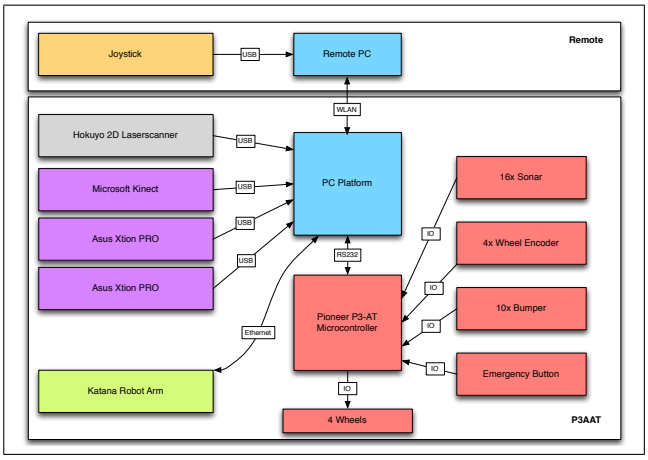


Fig. 2. Hardware Architecture.

A. Base Plattform Pioneer P3-AT

The robot, as seen on Figure 1, is based on the platform Pioneer P3-AT. The Pioneer 3-AT is a cost effective skid-steer four-wheel drive robot platform for research. The P3-AT carries 3 swappable batteries to supply the the entire system with power. The P3-AT reaches speeds of up to 0.8 meters per second and is able to carry a payload up to 12 kg. P3-AT comes with 8 sonar sensors and 5 bumpers on the back and on the front of the vehicle. It uses 100 tick encoders with inertial correction recommended for dead reckoning to compensate skid steering. The P3-AT has a user control-panel to access its micro-controller (Renesas SH2). The user control-panel has several buttons, LED's and interfaces: a motor enable pushbutton, a system reset, 2 AUX power switches, a battery charge indicator, a main power indicator and a MIDI programmable piezo buzzer. The micro-controller Renesas SH2 runs the ARCOS operating system for low level control of the engines, the sensor data readings and the battery level. The micro-controler operates at 44 MHz and features 32KB of RAM memory and 128KB of flash memory. It has a RS-232 serial port to communicate with the PC-platform. The PC-platform consists of a powerful 2.5GHz Intel i5 quad core CPU, 4GB of RAM, a 120GB SSD drive and an additional PCI-E FireWire card.

B. Robotic Arm: Neuronics Katana 450 6M90B

In order to manipulate the bottle position a robotic arm is used. The Neuronics Katana 450 6M90B is a five axis robotic arm capable of lifting a payload of 400g. The robotic arm is equipped with a mechanical angled gripper, without any additional sensors. The Katana 450 provides an Ethernet, USB and an IO interface [15]. The P3AAT communicates with the robotic arm via the Ethernet interface.

C. Sensor System

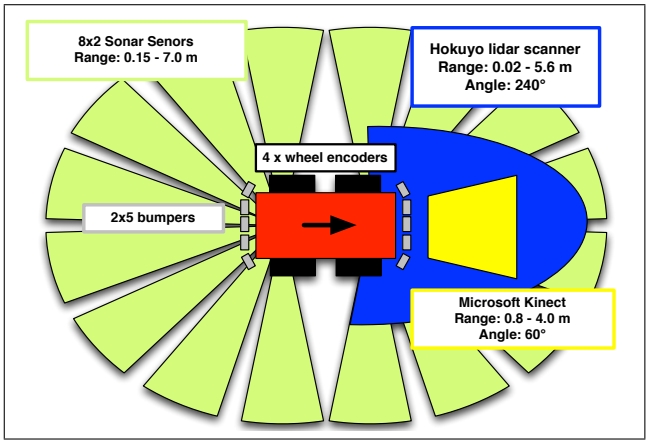


Fig. 3. Coverage of different sensor systems.

As shown in Figure 3 the robot is equipped with different kinds of sensors. The advantage of using multiple kinds of sensors is a more reliable environment measurement by fusing different sensors or selecting a sensor that matches in certain situations best. The red rectangle represents the P3AAT robot. The 16 green circle segments are representing the sonar sensors. The sonar sensors have a field of view of 360° and can be used for driving backwards.
The Hokuyo URG-04LX laserscanner has in theory a view range about 240°, due to the hardware construction the laserscanner has a field of view of 180°.
The Asus depth sensor consists of an IR camera and IR projector. It is using a structured light approach called light coding to generate respective 3D point clouds. The Microsoft Kinect has additionally a RGB camera and a multi-array microphone.

As mentioned before, the laserscanner is replaced by two depth sensors. One depth sensor is mounted slightly above the floor in the front area of the robot. The range of the depth sensors are about 0.8 to 4.0 meters. The consequence of this range is that near obstacles cannot be detected by the depth sensor. To solve this problem a second depth sensor was installed. This depth sensor is mounted on the upper frame of the robot, the view of this sensor is directed downwards in front of the robot. So near obstacles- in the region less then 0.8 meters can also be detected, because the navigation stack is able to manage different input streams.

The third depth sensor is used to detect the bottle. Therefore it is mounted on the upper frame of the robot and the view of the sensor is directed to the area in front of the robot.

III. Software

The P3AAT uses the Robot Operating System (ROS) as its operating system. ROS provides libraries and tools to help software developers to create robot applications. It provides hardware abstraction, device drivers, visualizers, message-passing and package management. ROS is licensed under an open source, BSD license [3].

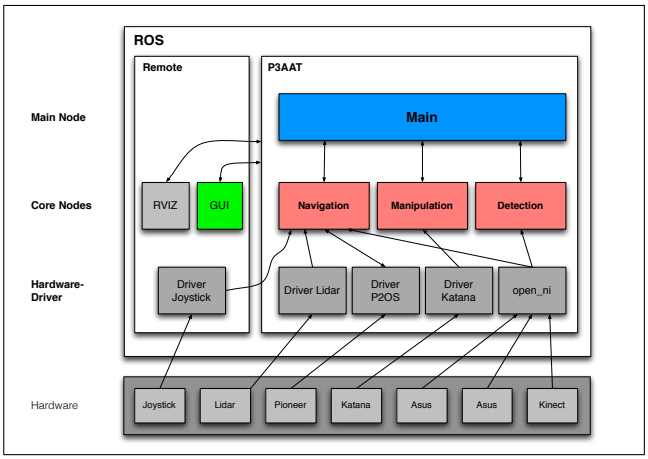


Fig. 4. Software Archidecture.

Figure 4 shows the distributed software architecture of the robot P3AAT. The communication between the nodes is handled by the Robot Operating System. Therefore the architecture is modular and scaleable. Most of the nodes are designed to run on the mobile robot platform. Only the nodes required for visualization purposes and remote control are launched on the remote PC. The Graphic User Interface

is designed to control the robot P3AAT. It is possible to start the autonomous operation mode, navigate to a specific point, start the bottle detection sequence or move the robot arm. Furthermore sensor readings and logging-messages are displayed in separate tabs. This sensor readings and other data is also displayed in the visualization tool RVIZ. Also the joystick driver is running on the remote PC, where the joystick is mounted.

On the remote PC the architecture is divided in three software layers. The lowest layer is the hardware driver. This nodes handle the communication with the hardware components. These components are provided by different communities. The driver-node P2OS is provided by the manufacturer Pioneer. The driver-node KNI is provided by the University of Osnabrueck. The other driver nodes are developed by the open source community.
The two other layers (main-node, navigation-node, manipulation-node and detection-node) are described in detail in the following Chapters.

A. Main-Node

The main-node represents a finite state-machine. Depending on the actual step the main-node invokes remote methods on the navigation, manipulation or detection-nodes.
The standard process is as follows: After the initial state the main-node invokes a method on the navigation-node to drive to a specific point - for example in front of the desk, where the bottle is located. After reaching this position a method on the detection-stack is invoked to detect the position of the bottle. Then the robot drives to the close vicinity of the bottle. Then a method on the manipulation-node is called to grip the bottle.

B. Navigation-Node

The navigation node is responsible for navigation. To navigate reliably in indoor environments the robot has to know its exact location. So for position estimation the P3AAT follows two approaches:

- The SLAM-Algorithm (Simultaneous Localization and Mapping) or GMapping, is a highly efficient Rao-Blackwellized particle filer to learn grid maps from laser range data [4], [5], [6], [7].
- The AMCL-Algorithm (Monte Carlo Localization), which uses a particle filter to track the current position of the robot against a known grid map [8], [9].

The navigation-node provides two interfaces for the main-node and the GUI. It is possible to navigate the P3AAT to a specific position in a known environment. This position can be in front of a desk, where the bottle is located.
Furthermore the P3AAT is able to look for the bottle. For this operation mode a basic search function was implemented. The robot is able drive to a random and unknown position. When the robot has reached this random position, the detection-node will be activated to find the bottle. If the bottle cannot be found, the robot drives to the next unknown position. In order to compute the random goal position a costmap grid is used. The costmap takes the sensor data from the robot and builds a 2D occupancy grid. The lower the value of a grid cell, the lower is the probability of a collision with an obstacle in the environment. The obstacles and its costs are marked with a high value in the costmap [10]. The path planning is carried out from the move-base node, which is developed by Eitan Marder-Eppstein. The move-base node links together a global and local planner to accomplish its global navigation task. The global planner provides a fast interpolated navigation function on a costmap to find a minimum cost plan from a start point to an end point in a grid. The local planner provides implementations of the Trajectory Rollout and Dynamic Window approaches to local robot navigation on a known map. Given a plan to follow and a costmap, the controller produces velocity commands to send to a P2OS-node, which sends commands directly to the P3AAT [11].

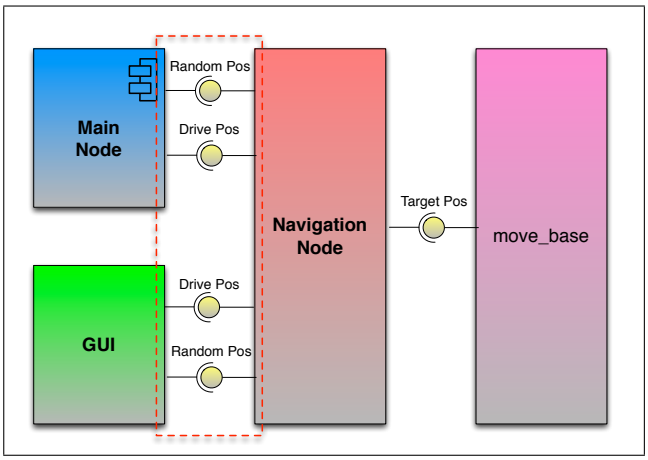


Fig. 5. Navigation Software Architecture.

C. Detection-Node

The detection node is designed to detect predefined bottles on a kind of table scene. The algorithm for detection is based on the functionalities of the Point Cloud Library (PCL) [12]. PCL provides a rich tool set for processing large 3D point clouds. The sensor used for object detection is the Asus Xtion Pro camera mounted on the upper frame of the robot. The object detection is responsible for the detection of possible objects within a scene, checking for matching objects, calculating the position and pose of matching objects and forwarding the result to the next processing step.

The detection itself consists of two different stages. A so called Training-stage and a Detection-stage. In the first stage a set of descriptors from objects will be created, which are used in the second stage for recognition. As descriptor/signature for recognition, Viewpoint Feature Histograms (VFH) are used. VFH is a descriptor for 3D point cloud data that encodes geometry and viewpoint [13]. Furthermore it is also supported by the PCL. During the Training stage these VFH descriptors are created for the respective bottle from different angles and distances.

The architecture of the Detection Service is illustrated in Figure 6 and consists of the following three nodes:

- Detection Node
- Segmentation Node
- Recognition Node

The Detection Node is responsible to coordinate detection callups from the P3AAT Main Node or the User Interface. First the segmentation service is called and after receiving possible objects the recognition service is started. Therefore it is also easy to change/replace the segmentation or recognition services by other approaches if needed. Finally, the Detection Node returns the position of a matching object. The Segmentation pipeline fetches first a point cloud from the camera and starts to detect the largest plane model within the cloud. From the convex hull of the plane the points lying on the plane are extracted and clustered. If no cluster objects are found the next largest plane is determined and processed. Furthermore segmentation calculates the center points of the detected cluster surfaces.

The Recognition Node receives possible cluster objects for further processing. First the VFH descriptor is computed for each cluster and is used in order to match between previously recorded VFH descriptors, which are loaded into the Recognition Node during startup.

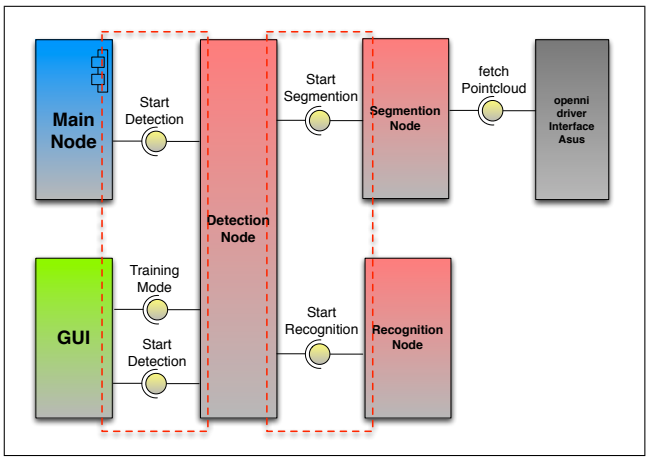


Fig. 6. Detection Software Architecture.

D. Manipulation-Node

The Katana 450 provides a KNI (Katana Native Interface) to control the robot arm, using the C++ programming language. This interface is used for the ROS integration of this robotic arm. The software module responsible for object manipulation is using ROS to move the robotic arm. The manipulation-node is designed in a service oriented approach, by using ROS Services [14]. The software architecture of the manipulation-node is shown in Figure 7 and provides following services:

- Pickup bottle: This is the main use case of the manipulation-node. The node receives the position of the bottle, calculates the trajectory, performs the inverse kinematic and moves the Katana robotic arm, by using the Katana stack, towards the bottle position. Afterwards the gripper is closed and the robotic arm is moved back to the home position. The service response contains information if a bottle was gripped or if any error occurred.
- Operating Mode: This service is used to switch the operating mode. The service Pickup bottle can only be used in automatic mode and all other services only in manual mode. This prevents some manual intervention during automatic mode.
- Move: The move service provides the ability to drive the Katana robotic arm by using joint positions as well as the position of the end-effector.
- Gripper: The gripper service can be used to open or close the gripper mounted on the robotic arm. The mentioned services are mainly used by the main-node, but they can be called by any other ROS nodes such as a test node or the remote GUI as well.

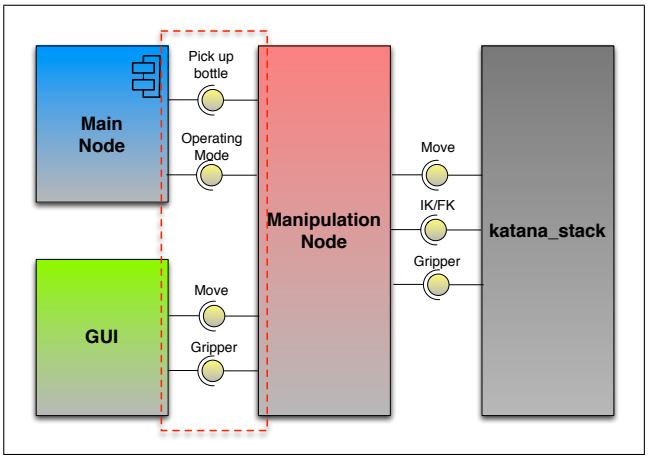


Fig. 7. Manipulation Software Architecture.

To be able to pick up the bottle, the main-node receives the target position of the detected bottle and calculates a trajectory from the robots home position to the position of the bottle. Two different trajectory modes have been evaluated which are demonstrated in Figure 8. The black trajectory mode is first adjusting the height of the robots tool point to the height of the bottle grasping point. Afterwards the robot is moving its open gripper towards the bottle. The red trajectory follows a much simple approach by calculating a direct three dimensional line from the robot's current position to the bottle position. The first trajectory mode is more capable if the bottle position is on a table and higher than the robot's current position.

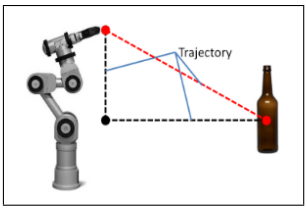


Fig. 8. Trajectory modes of the robot arm.

IV. Evaluation

During the software integration and testing phase some structural weaknesses has been discovered.

Due to its mechanical construction, the P3AAT is only able to detect a bottle in a distance of about 1,5 meters. The room coordinates of the bottle position are transformed into world coordinates. After that the robot drives towards the bottle. At this step the Asus camera is not able to detect the bottle any more, since it is out of focus. This means that the Katana robotic arm tries to grasp the bottle at a position, which was calculated from a distance of about 1,5 meters. Any unknown deviation while navigating towards the bottle position or before starting the grasping therefore leads to an unsuccessful or imprecise grab of the bottle. Additional sensors mounted directly on the robotic arm or a movable camera are able to detect the bottle within the grab area of the arm. This will increase the accuracy of detecting the bottle position.

During the testing phase we tried to optimize the cycle time. A performance optimization in the detection process has been carried out by restricting the view of the camera to a smaller field of interest. The result was a reduction of cycle time by three seconds.

V. Conclusion

The mobile robot P3AAT project successfully demonstrated the integration of autonomous navigation, bottle detection and grasping into a mobile platform. The chosen service and node oriented software architecture based on the Robotic Operating System clearly showed the advantages of a modular system. So the software- and the hardware-system can easily be customized or extended by future student projects.

References

[1] G. van den Broek et al., Ambient Assisted Living Roadmap, VDI/VDE-IT, 2009.

[2] J.C. Augusto et al., Handbook of Ambient Assisted Living - Technology for Healthcare, Rehabilitation and Well-being, IOS press, 2012.

[3] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, Andrew Ng. "ROS: an open-source Robot Operating System", Retrieved 3 April 2010

[4] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II. Robotics and Automation Magazine, 13(3):108117, 2006.

[5] H. Durrant-Whyte and T. Bailey. Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. Robotics and Automation Magazine, 13(2):99110, 2006.

[6] S. Thrun. Robotic mapping: a survey, Exploring artificial intelligence in the new millennium, 2003.

[7] R. Patrick Goebel, ROS By Example - Volume 1: A Do-It-Yourself Guide to the Robot Operating System, Pi Robot Production, pp. 90 - 94, 2012.

[8] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. Markov localization for mobile robots in dynamic environments. Journal of Artificial Intelligence. 11, 1999.

[9] R. Patrick Goebel, ROS By Example - Volume 1: A Do-It-Yourself Guide to the Robot Operating System, Pi Robot Production, pp. 95 - 106, 2012.

[10] Joon-Hong Seok, Joon-Yong Lee, Changmok Oh, Ju-Jang Lee and Ho Joo Lee. Diverse Multi-Path Planning with a Path-Set Costmap, 2011.

[11] R. Patrick Goebel, ROS By Example - Volume 1: A Do-It-Yourself Guide to the Robot Operating System, Pi Robot Production, pp. 29 - 46, 2012.

[12] Radu B. Rusu. and Steve Cousins. 3d is here: Point cloud library (pcl). In 2011 IEEE International Conference on Robotics and Automation (ICRA), pages 14 -, May 2011.

[13] Radu B.Rusu, Gary Bradski, Romain Thiboux, John Hsu. Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram. In 2010 IEEE International Conference on Intelligent Robots and System (IROS), pages 1-4, 2010

[14] Sebastian Thrun, Wolfram Burgard, and Dieter Fox, Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press, 2005.

[15] Neutronics AG. Katana 450 User Manual, 2001-2008, Document Number 233493, Version 2.0.4, page 7, 12

[16] Stanford University, Willow Garage, Univerity of Southern California ROS: an open-source Robot Operating System, page 3

Automatic Modelling and Observers Generation for Model-Based Diagnosis System for ROS-Based Robotic Systems

Safdar Zaman, Gerald Steinbauer

***Abstract*— Fault detection and repair are necessary actions for a robotic system that claims to be fully autonomous. Model based diagnosis system compares *diagnosis model* with the *observed model* in order to detect faults at runtime. *Diagnosis model* is a set of rules describing correct behavior of the robotic system while the *observed model* is the output of the monitoring system carried out by several observers with different parameters. In the contribution we present a methodology for automatic generation of the diagnosis model and diagnosis observers. The methodology gets the running system as input and extracts required entities like $nodes$ and $topics$. The work also presents the calculation of the node related properties like cpu and $memory$ usage, and also calculation of properties like $frequency$ and $deviation$ related to each topic. For testing purpose we used Mapping scenario with *Pioneer-3DX* robot. Methodology successfully generates diagnosis model and observers with required parameters. the presented work offers two major contributions: automatic generation of diagnosis model, and generation of observers with required system parameters.**

Key–Words: **Model-Based Diagnosis, Modelling, Observers, and Learning.**

I. Introduction

For achieving a given specific task automatically, every robotic system uses a number of sensors like Laser sensor, Camera, Gps, Imu, .etc. During accomplishing its task, it is likely that robotic system gets faults if a sensor leaves functioning or starts publishing wrong data because of malfunctioning of some other component. To cope with this problem it is necessary to have a monitoring system that can not only detect such faults but also can repair them at run time. In order to acquire the model of the behavior of the robotic system one has to have some mechanism to analyse the the data coming from its sensors. This data describes the behavior of that robotic system at runtime and can be used to model that behavior. The model of the correct behavior can be obtained from robotic system when it does not generate any faults at runtime. A model of observed behavior of the robotic system is one which is obtained by monitoring the data from its sensors at runtime. These both models can then be used for fault detection. For a robotic system to have a self-monitoring capability at runtime, it is important for it to know its correct behavior besides having obtained observed behavior. A fault can be detected if there is a significant deviation between correct and observed behavior of the robotic system. The Model-Based Diagnosis (MBD) system uses *diagnosis model* in order to diagnose and detect faults. This *diagnosis model* is a set of rules representing the correct behavior of the robotic system. A more carefully built *diagnosis model* leads to more correct and effective fault detection process. Previously diagnosis model was prepared by the user and MBD system used it for fault detection. For writing diagnosis model user had to closely look into the structure of the robotic system and communication between its executing components. Generating *diagnosis model* by hand may lead to incorrect, less saturated, and incomplete model. Moreover, manually generating process of the diagnosis model is more time consuming, hard, and more attention requiring especially when robotic system is more complex and contains a lot of communicating nodes. In order to overcome all of these problems an automatic generation of *diagnosis model* has become vital because it is flexible, less time consuming, more correct, easily repeatable, and finally machine oriented. Automatic generation of the observers is as important as generation of *diagnosis model* for the reason that observers need parameters like *CPU* and *Memory* usage for nodes, and frequency and deviation parameters for publishing topics, .etc. All these parameters can be easily calculated automatically during generation of *diagnosis model* parallely. In the contribution we present a methodology that basically performs three important tasks: automatic generation of *diagnosis model*, calculation of the parameters, and finally generation of the observers. Generated *diagnosis model* and observers are used by MBD system to find out root cause of the faults if any. The presented work is ROS(Robot Operating System)-based that means it revolves around the concepts of $nodes$, $topics$, $messages$ and $services$ [1].

The presented work is organized as follows: Related work on software, and hardware fault detection and repair is reviewed in Section II, it also includes learning of fault detection models from communication data in robotic systems. Section III explains our MBD system and its modules like *observers*, *diagnosis model server*, *diagnosis engine*, and *diagnosis repair engine*. MBD system is in fact the prerequisite of the presented work. MBD system uses diagnosis model and observers for fault detection and afterwards repairs the detected faults using diagnosis repair engine. In section IV the methodology of the presented work is explained in detail. Section V describes the experiment for mapping scenario where we use *Pioneer DX-3* robot with *SICK* laser and *IMU* sensors as shown Figure 4. Section VI briefly summarizes the presented contribution and provides future challenges.

S. Zaman and G. Steinbauer are from the Institute for Software Technology, Graz University of Technology, Graz, Austria. {szaman,steinbauer}@ist.tugraz.at

II. Related Work

Research on fault detection and repair has been under consideration of the researchers in the direction of both software and hardware. Early research works like [2] presented mechanism for fault detection for robot control software based on *MIRO* framework [3]. Our approach uses the ROS framework [1] which is new, rapidly growing and mostly used by the robotic laboratories. Automated learning of the communication models for robot navigation software is presented by [4], this work describes an approach that derives a model of the communication behavior within control software. Some works presented approaches for repairing the faults after detecting them [5]. It also provides mechanisms for reconfiguration of the mobile robot at runtime. It provides a control framework which is capable of reconfiguring the functionality of hardware drive units and handles their geometries. The work [6] provides a ROS-based architecture for the fault detection and repair in both software and hardware. Faults dealt in this work includes getting down a running software component, not providing sensor data with required frequency and so on. On-line diagnosis system for fault detection in current measurements in a distributed embedded hybrid system like Xerox DC265 printer has been presented in previous works [7], [8], which provide means towards hardware fault detection. Works like [9], and [10] presented some learning techniques for the model of communication data in robotic systems. These provides techniques for creating a statistical model out of internal data exchange and communication in the robotic system.

III. Prerequisite System

Presented work is an extension of our Model Base Diagnosis (MBD) system [11]. It is Robot Operating System (ROS) based architecture. MBD system has different components: a set of observers, one diagnosis model server, one diagnosis engine, and one diagnosis repair engine. Observers include general observer (GObs) for observing the topic's frequency, node observer (NObs) for monitoring the node if it is running or not, diagnostic observer (DObs) for integrating existing ROS */diagnostics* topic, hardware observer (HObs) for observing the diagnostic board, qualitative observer (QObs) for measuring the qualitative trend of a value inside the message coming on a topic, multiple observer (MObs) is to observe the conditional communication between the topics, and finally property observer (PObs) for checking the properties like cpu and memory usage for a node. An observer is an executable software entity functioning as ROS node to publish information related to the system components on ROS topic */observations*. This information is in the form of First Order Logic (FOL) sentences like $ok(topic\_name)$ or $\neg running(node\_name)$ describing that topic $topic\_name$ is working properly and node $node\_name$ is functioning abnormally. Diagnosis model server is a ROS action server that reads diagnosis model of the robotic system from a *YAML* file and whenever required it publishes this diagnosis model in a required form for the diagnosis engine. Figure 1 describes the MBD system with its all modules:

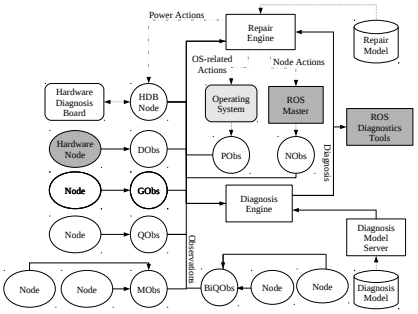


Fig. 1. Architecture of the Model Based Diagnosis System showing its modules and communication between them.

Diagnosis engine follows the diagnosis principles presented in [12] and finds the diagnosis at runtime by using diagnosis model and observations coming from the observers. Diagnosis engine publishes diagnosis on ROS topic */diagnosis* in the form of set $bad(m)$ for faulty components m, and $good(m)$ for the working components m. After having computed diagnosis MBD system activates Diagnosis Repair Engine which is in fact a planner based repair engine using well known Planning Domain Definition Language (PDDL) [13] for finding set of action plans for repair. Diagnosis repair engine takes diagnosis and observations, and invokes the planner to get the action plans for the repairing. After getting action plans diagnosis repair engine invokes action servers like $start\_node(node)$ or $stop\_node(node)$ to start or stop a software node, and action servers $power\_up(device)$, and $shutdown(device)$ in order to switch off or on a hardware device. During this whole process of fault detection and repairing MBD system is always active and continuously observes the robotic system at runtime. Our MBD system can be downloaded from the ROS Wiki[1].

The Model-based diagnosis system revolves around the diagnosis model and observers. Diagnosis model provides predefined model while observers provide the currently observed model, on the basis of these both models MBD system detects the faults, therefore it is very much important to generate these both entities correctly.

IV. Presented Methodology

In the contribution we present a methodology for automatic generation of diagnosis model and observers for the MBD system. A special software entity called $Generator$ has been written to accomplish these tasks. Diagnosis model is used by the diagnosis engine for detecting the faults whereas observers are used for monitoring the robotics system at runtime. Following subsections describes the modules of presented methodology:

A. Diagnosis Model

Diagnosis model is a central concept in the Model based diagnosis system. Diagnosis model not only describes the correct behavior of the running robotic system but it also

[1] http://www.ros.org/wiki/tug_ist_model_based_diagnosis

plays vital role in detecting the root cause of the failure. The precisely and more carefully created diagnosis model is very much important for the fault detection process of diagnosis system. Process of creating model manually can lead to less precise and less system describing model because user needs to carefully describe the components of the system and association between them which is liable to error. Therefore this work presents automatic generation of the diagnosis model instead of creating it manually. Following two subsections describe the structure and automatic generation the diagnosis model.

B. Diagnosis Model Structure

Diagnosis model is a structure that defines five components: (1) a string denoting the proposition that represents the AB (abnormal) predicate, (2) string representing $\neg AB$ (normal), (3) a string for a prefix denoting a negative literal, (4) set of propositions, (5) set of clauses defining diagnosis model. Following is a simple diagnosis model structure showing a scenario of running node "$node$" which publishes data on a topic named "$node\_topic$". This structure defines three strings "AB","NAB", and "$not\_$" , set of propositions for the correct status of nodes and topics, and finally set of rules defining behavior of the communicational nodes. The rules describe that a node is fine if it is running and its topic publishes data with the right frequency. This is described with a simple structure of diagnosis model as below:

```
ab: "AB"
nab: "NAB"
neg_prefix: "not_"
props:
  - running(node)
  - ok(node_topic)
rules:
  - NAB(node) -> running(node)
  - NAB(node) -> ok(node_topic)
```

C. Automatic Generation of Model

The process of automatic generation of the model assumes that robotic system during the training runs correctly and no fault occurs during this training. This process is carried out by a software entity called $Generator$ which is now a part of MBD system, and is described in the Figure 2. In order to keep this model generation process flexible and less complex it is divided into different modules. Each module has a specific task where it takes some input and produces the output which is further input for the next module. Overall input of the generation process is currently executing robotic system, and its output is automatically generated observers and diagnosis model. Following subsections describe each of the modules in detail:

1) Get System State: This sub phase takes internal representation of the currently running robotic system from the ROS master. This internal representation is the system state of the currently running robotic system. It includes all subscribers, publishers, topics, services, .etc. At runtime, this phase is executed repeatedly and concurrently to find out if a new new node is started running. Topic is the means on which nodes share information among them and it provides an interface between the nodes for

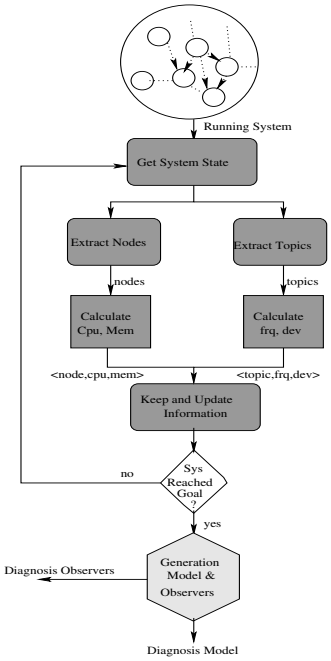


Fig. 2. Generator: Overview of the process of automatic modeling and observers generations.

communication. System status may also be considered as a graph representation of the nodes and topics where nodes make the "nodes" of the graph and topics are on the "links" between the nodes. The system state looks like list of lists of components of the system. Basically three lists; 1) list of publishers giving nodes which publish on topics, 2) list of subscribers that receive information from the topics, and 3) the list of services that provide some functionality when requested. System state can be described in terms of set S as follows:

$S = \{Pb, Sb, Sr\}$
Where Pb, Sb and Sr are the sets ROS publishers, subscribers, and services as:

$Pb = \{Pb_1, Pb_2, Pb_3, ...\}$ where $Pb_i = \{t_i, Pb_{t_i}\}$ and Pb_{t_i} is the set of all publisher nodes for the topic t_i.

$Sb = \{Sb_1, Sb_2, Sb_3, ...\}$ where $Sb_i = \{t_i, Sb_{t_i}\}$ and Sb_{t_i} is the set of all subscriber nodes for the topic t_i.

$Sr = \{Sr_1, Sr_2, Sr_3, ...\}$ where $Sr_i = \{s_i, Pr_{s_i}\}$ and Pr_{s_i} is the set of all service provider nodes for the service s_i.

2) Extract Nodes: Main component of a ROS-based robotic system is the $node$. A $node$ is an executable program inside the system which transmits information, communicates with other system components. A node can be either $publisher$ due to transmitting information, or $subscriber$ due to receiving information from other nodes. The node

has very important influence over the running system performance, therefore this module takes the system status as input and finds all executing nodes of the system. One node could appear more than one time in the system status if it is publisher of many topics. This sub phase also takes care of repeating nodes and makes a globe list of the nodes where one node appears only once and it is updated on each iteration if a new node comes in. This global list of nodes can be accessed within whole generator.

3) Extract Topics: The second important component of a ROS-based robotic system is the $topic$. Topic is a channel on which $node$ publishes its information to be further used by other nodes. This sub phase extracts the topics from the system status and creates a globe list of the topics where a topic appears only once. In the system status it could be the case that more than one node can publish information on the same topics. The global list of the topic is accessible within the whole generator.

4) Calculate Frequency and Deviation for Topics: Information on a topic appears with some frequency. This sub phase creates a callback against each publishing topic and calculates current frequency and deviation of appearing data on the topic. It is done by calculating the time of arriving data on a topic. The Δt is the time difference between two occurrences, this is also calculated and filled into a circular queue as shown in the following equations:

$$\Delta t_i = curr\_time_i - prev\_time_i$$

$$T = \frac{1}{N} \sum_{i=1}^{N} (\Delta t_i) \quad (1)$$

$$frq = \frac{1}{T} \quad (2)$$

The $curr\_time_i$ and $prev\_time_i$ are the current and previous time at occurrence number i. N is the number of occurrences occurred within time ws which is time window size. Equation 1 calculates average time differences between all successive occurrences happened in the time ws, while Equation 2 calculates the frequency. Figure 3 gives an idea of occurrences of data and how window slides along it to get the calculations for frequency. Each time a new occurrence comes the frequency is calculated by dividing 1 by average of all the values in the queue. The average is calculated by summing up all the delta times in the queue and dividing it by the number of occurrences obtained during time window size ws. This module also calculates properties like cpu and memory usage for each running node. This is calculated by getting process identification *pid* of a node from the *ROS Master*, and this *pid* is then used in sub process in Linux shell to get the cpu and memory usage in percentage for the node bearing *pid*. Cpu and memory usage is calculated during whole run and the maximum usage of cpu and memory is considered for the observation by the observers.

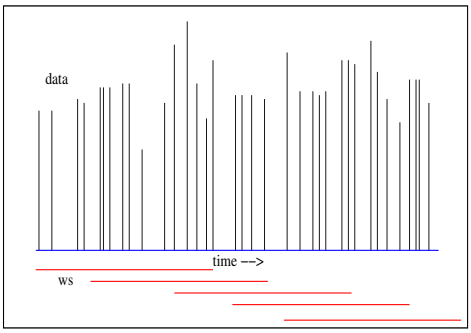


Fig. 3. Occurrence of data with sliding window for calculations.

5) Storing extracted information: The information extracted and calculated by above sub phases is kept and repeatedly stored in a two data structures each for node and a topic. Both node and topic data structures are shown below:

```
node_data_structure:{
                      node_name,
                      node_pid,
                      max_cpu,
                      max_mem,
                      pub_topic_list,
                      sub_topic_list
                    }

topic_data_structure:{
                      topic_name,
                      publish_frq,
                      publish_dev
                    }
```

Node data structure contains name, process id, maximum cpu and memory usage, list of published topics, and list of subscribed topics by each node. Topic data structure maintains topic name, publishing frequency and deviation for each topic in the system. All this information is stored for later usage for automatic generation of the model and observers.

6) Observers Generations: The above process continues as long as robotic system task is achieved like during *Mapping* moving to a point 'A' and then coming back. All the necessary requirements for the observers are calculated during this whole process. There are different observers with different requirements like node observer (NObs) needs name of the node, property observer (PObs) needs name of the node, name of the property (cpu or mem) and then maximum value, general observer needs topic name, required frequency, deviation, window size and slope. All other observers also have some requirements. This sub phase retrieves all the collected information during the robotic system run, and makes a launch file preparing all the observers with their needed parameters. This launch file can be used to bring all the observers up.

7) Model Generation: This sub phase is generates diagnosis model by taking robotic system as input. It retrieves all the collected information during the above sub phases like list of nodes, topics, subscribing topics and publishing topics. In the current generation of the diagnosis model we consider only following three clauses to make the rules for the model:

1) for each $n \in N$ add:
   $\neg AB(n) \rightarrow running(n)$
2) for each $n \in N$ and each $t \in out(n)$ add:
   $\neg AB(n) \wedge \bigwedge_{i \in in(n)} ok(i) \rightarrow ok(t)$

3) for each $n \in N$ and each $p \in property(n)$ add:
$\neg AB(n) \wedge running(n) \rightarrow ok(n, p)$

Where N is the set of all ROS nodes, $out(n)$ is the set of all publisher (outgoing) topics of node n, likewise $in(n)$ is the set of all subscribing (incoming) topics to the node n. The function $property(n)$ is the set of properties utilized by the node n, e.g. $cpu, memory$.
Applying the above three rules a diagnosis model is automatically generated for the running system. This automatically generated diagnosis model is afterwards stored in a YAML file for subsequent use in MBD system.

V. Evaluation

For evaluating *Generator* we carried out experiment for the scenario of mapping with *Pioneer DX-3* robot equipped with *SICK* laser sensor as shown in Figure 4.

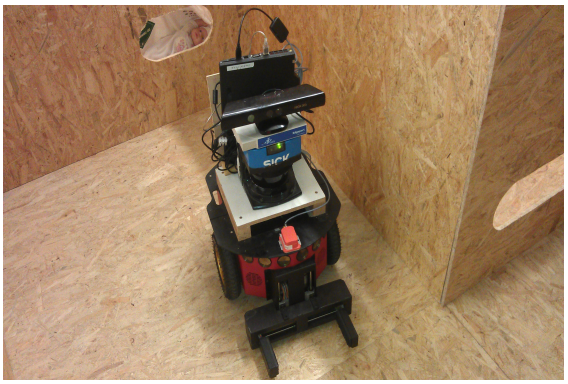


Fig. 4. Pioneer DX-3 robot with SICK laser sensor for mapping.

Presented teleoperated mapping scenario needs ROS nodes namely *aria* for the pose estimation and movement of the robot, *sicklms* node for Sick laser sensor and *gmapping* for generating the map, *telop_node* for teleoperating the robot, and *joy_node* for the joystick. Node *aria* publishes robot odometry on topic */pose*, *sicklms* node publishes laser scans on topic */scan*, and *gmapping* constructs the map and publishes it on the topic */map*, *telop_node* node provides command velocity to the robot base on topic */cmd_vel*, and *joy_node* provides connection to joystick and publishes signals on topic */joy*. Both nodes *aria* and *sicklms* need no subscribing topic while *gmapping* subscribes two topics */pose* and */scan*. Basically we have two sets for nodes $N = \{aria, sicklms, gmapping, telop\_node, joy\_node\}$ and for topics $T = \{pose, scan, map, cmd\_vel, joy\}$ This scenario is shown in the Figure 5.

The automatically generated diagnosis model by the *Generator* for this mapping scenario is described below:

```
ab: "AB"
nab: "NAB"
neg_prefix: "not_"
props:
  - running(aria)
  - running(sicklms)
  - running(joy_node)
  - running(telop_node)
  - running(gmapping)
  - ok(pose)
  - ok(scan)
  - ok(map)
```

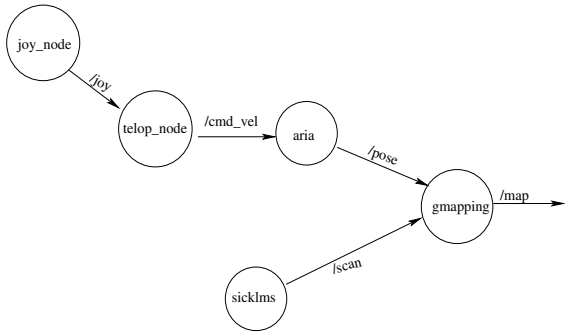


Fig. 5. Teleoperated mapping scenario.

```
  - ok(joy)
  - ok(cmd_vel)
  - ok(aria,cpu)
  - ok(aria,mem)
  - ok(sicklms,mem)
  - ok(sicklms,cpu)
  - ok(joy_node,mem)
  - ok(joy_node,cpu)
  - ok(telop_node,mem)
  - ok(telop_node,cpu)
  - ok(gmapping,mem)
  - ok(gmapping,cpu)
rules:
  - NAB(aria)->running(aria)
  - NAB(aria),ok(cmd_vel)->ok(pose)
  - NAB(aria),running(aria)->ok(aria,cpu)
  - NAB(aria),running(aria)->ok(aria,mem)
  - NAB(laser)->running(laser)
  - NAB(sicklms)->ok(scan)
  - NAB(sicklms),running(sicklms)->ok(sicklms,cpu)
  - NAB(sicklms),running(sicklms)->ok(sicklms,mem)
  - NAB(gmapping)->running(gmapping)
  - NAB(gmapping),ok(pose),ok(scan)->ok(map)
  - NAB(gmapping),running(gmapping)->ok(gmapping,cpu)
  - NAB(gmapping),running(gmapping)->ok(gmapping,mem)
  - NAB(joy_node)->running(joy_node)
  - NAB(joy_node)->ok(joy)
  - NAB(joy_node),running(joy_node)->ok(joy_node,cpu)
  - NAB(joy_node),running(joy_node)->ok(joy_node,mem)
  - NAB(telop_node)->running(telop_node)
  - NAB(telop_node),ok(joy)->ok(cmd_vel)
  - NAB(telop_node),running(telop_node)->ok(telop_node,cpu)
  - NAB(telop_node),running(telop_node)->ok(telop_node,mem)
```

The diagnosis model shown above comprises of twenty propositions and twenty rules. Propositions are for the correct status of all five nodes and their five topics. Rules regarding node $aria$ describe that the node is ok if it is running, its input topic is publishing properly, it is running under cpu and memory limits. The same rules are for all five nodes.

Generation of the observers is also carried out at the end. All the acquired information from the running system is used to make the parameters for the relevant observers. Each topic gets one GObs, each node gets three observers NObs, PObs for cpu, and PObs for memory usage observation. The launch file for observers was automatically generated, three observers namely *NObs*, *PObs*, and *GObs* for node *aria*, sicklms, and topic *map* with extracted system parameters are shown below:

```
<node pkg="tug_ist_diagnosis_observers" type="NObs.py" name="ariaNObs" >
<param name="node" value="aria" />
</node>

<node pkg="tug_ist_diagnosis_observers" type="PObs.py" name="sicklmsCpuPObs" >
<param name="node" value="sicklms" />
<param name="property" value="cpu" />
<param name="max_val" value="2" />
<param name="mismatch_th" value="5" />
<param name="ws" value="150" />
</node>

<node pkg="tug_ist_diagnosis_observers" type="GObs.py" name="mapGObs" >
<param name="topic" value="map" />
<param name="frq" value="19.8117276201" />
<param name="dev" value="0.0298082886436" />
<param name="ws" value="100" />
</node>
```

The values of the parameters like max_val, frq, dev, are calculated automatically while some values like $mismatch\_th$

and ws are taken from the user. An observer issues an alarm when observed value mismatches the given property value continuously for times more than $mismatch\_th$. Alarm is also issued when the frequency of a topic goes beyond the given frequency frq with dev within window size ws. Such observers for all system nodes and topics are automatically generated. The current model contains five nodes and five topics so there are as many as twenty observers are generated among them five NObs observers, five GObs observers and two PObs observers for each node for cpu and memory usage.

VI. Conclusion and Future Work

In the contribution the presented work presents automatic modeling for the ROS based Model-Based Diagnosis (MBD) system. Diagnosis model is a structure that describes the correct behavior of the robotic system. MBD system uses diagnosis model for the fault detection. Carefully and correctly created diagnosis model is necessary for precise, effective and efficient fault detection process. This work offers a $Generator$ that has been created as part of MBD system for the automatic generation of the diagnosis model. At runtime the $Generator$ takes the robotic system as input and finds system components, association between them, which component sends message to which component, what properties like cpu and $memory$ usage for each component; all this information is automatically extracted from the system. Some rules have been defined which are applied to automatically generate the diagnosis model. Besides generation of the diagnosis model, $Generator$ also generates observers namely Node observer (NObs) for each node, General observer (GObs) for each topic, Property observers (PObs) to monitor cpu and memory for each node, are also automatically generated.

For future work we intend to consider more practical and complex robotic system for automatic modelling. It will also be needed to investigate how robust the model generation is. More rules for conditional communication need to be added to deal with triggering nature of communication between the topics. Future work is also to add more rules and put more intelligence for automatic modelling. It is also intended for future work to use automatically generated model along with diagnosis engine of MBD system to validate the generated model. Also future work includes applying some learning techniques to learn the correct behavior of the system.

VII. Acknowledgment

Safdar Zaman gratefully acknowledges the support from Higher Education Commission (HEC) of the government of Pakistan funding his PhD studies at Graz University of Technology in Austria.

References

[1] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software in Robotics*, 2009.

[2] Gerald Steinbauer, Martin Mörth, and Franz Wotawa. Real-time diagnosis and repair of faults of robot control software. In *International RoboCup Symposium*, volume 4020 of *Lecture Notes in Computer Science*, Osaka, Japan, 2006. Springer.

[3] Hans Utz, Stefan Sablatng, Stefan Enderle, and Gerhard K. Kraetzschmar. Miro middleware for mobile robot applications. *IEEE Transactions on Robotics and Automation, Special Issue on Object-Oriented Distributed Control Architectures*, page 18(4): 493497, August 2002.

[4] Alexander Kleiner, Gerald Steinbauer, and Franz Wotawa. Towards Automated Online Diagnosis of Robot Navigation Software. In *First International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR 2008)*, volume 5325 of *Lecture Notes in Computer Science*, pages 159–170. Springer, 2008.

[5] Mathias Brandstötter, Michael Hofbaur, Gerald Steinbauer, and Franz Wotawa. Model-based fault diagnosis and reconfiguration of robot drives. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[6] P. Lepej, J. Maurer, G. Steinbauer, S. Uran, and S. Zaman. An integrated diagnosis and repair architecture for ros-based robot systems. In *International Workshop on Principles of Diagnosis (DX 2012)*, Great Malvern, UK, 2012.

[7] F. Zhao, X. Koutsoukos, H.Haussecker, J. Reich, and P. Cheung. Distributed monitoring of hybrid systems: A model-directed approach. *International Joint Conf on Artificial Intelligence (IJCAI01)*, page Seattle, August 2001.

[8] X. Koutsoukos, F. Zhao, H.Haussecker, J. Reich, and P. Cheung. Fault modeling for monitoring and diagnosis of sensor-rich hybrid systems. *Proceedings of IEEE Conference on Decision and Control (CDC 2001)*, pages Orlando, FL, Dec, 2001.

[9] R. Golombek, S. Wrede, M. Hanheide, and M. Heckmann. Learning a probabilistic self-awareness model for robotic systems. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[10] R. Golombek, S. Wrede, M. Hanheide, and M. Heckmann. A method for learning a fault detection model from component communication data in robotic systems. In *Seventh IARP Workshop on Technical Challenges for Dependable Robots in Human Environments*, Toulouse, France, 2010.

[11] S. Zaman, G. Steinbauer, J. Maurer, P. Lepej, and S. Uran. An integrated model-based diagnosis and repair architecture for ros-based robot systems. In *IEEE International Conference on Robotics and Automation (ICRA-2013)*, Karlsruhe, Germany, 2013.

[12] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57 – 95, 1987.

[13] Craig Knoblock, Anthony Barrett, Dave Christianson, Marc Friedman, Chung Kwok, Keith Golden, Scott Penberthy, David E Smith, Ying Sun, and Daniel Weld. Pddl the planning domain definition language. *AIPS-98 Competition Committee*, 78(4):1–27, 1998.

Making Service Robots Safer - Affordable Tactile Sensing for Large Surface Areas

Michael Zillich, Walter Wohlkinger and Wendelin Feiten

Abstract— **Most of today's robots, while being equipped with arrays of ever more impressive sensors, still lack a fundamental ability so readily available to biological systems: a sense of touch. With the exception of a few humanoid platforms tactile sensing is typically limited to small areas such as palms or finger tips. In this paper we present a tactile sensing system designed specifically to meet the needs of safe service robotics for covering large, possibly curved surfaces of the robot. The sensors are at the same time highly sensitive and robust, cheap to manufacture and can be made to have almost any size and shape. As such the presented system can ideally complement existing systems and their high spatial resolution required in finger tips with a system providing highly sensitive and impact resistant coverage for the rest of the body.**

I. INTRODUCTION

Service robots operating in real world environments such as offices or home environments often need to navigate in tight spaces or in close vicinity to humans. And while sensing (especially 3D vision) is improving at a rapid pace, there will always be some obstacles that are hard to detect (tables with a glass surface, thin metal table legs) or are actually moving in unpredictable ways (such as pets or children). Think for example of a robot struggling a bit to navigate a doorway, and the impatient user tries to squeeze past the robot through the door, when the robot suddenly makes a corrective manoeuvre and pins the user against the door frame. Sonar and IR rings for close range collision detection have a very limited field of view (typically in one plane) and thus can not provide full coverage. Therefore we believe that safe operation of service robots requires tactile sensing for collision detection across large parts of the robot body.

Tactile sensing, however, is one of the big open areas in robotics. Many solutions target sensitive high resolution sensors for use in robotic fingers to support tasks such as grasp stability assessment [1] or tactile servoing [2]. These solutions typically use piezo-resistive or piezo-capacitive elements in a regular array on a printed circuit board. They are thus by design limited to small areas and tend to be rather fragile. A problem when covering larger areas beyond fingers is the amount of cabling required to get the large number of signals off the sensors. Multiplexing and serial bus systems can alleviate this problem to some degree, but the underlying problem remains, which is that not all parts of a robot body require the same high resolution and fidelity in tactile sensing. As a consequence comparatively little attention has been paid to covering large and possibly curved, oddly shaped areas with rugged sensors that are robust to impact and wear.

Michael Zillich and Walter Wohlkinger are with the Automation and Control Institute, Vienna University of Technology, Austria [zillich,wohlkinger]@acin.tuwien.ac.at, Wendelin Feiten is with Siemens Corporate Research, Munich

In this paper we propose to revisit an old principle for usage in a sensing system specifically addressing the need to cover large areas of a robot, possibly the whole chassis. The basic principle is used in a coarser version in crude push buttons where a pressure sensor inside an airtight rubber tube detects the increase in pressure as the user hits the device. In the presented system a core of soft foam rubber with an embedded pressure sensor is covered in an airtight sleeve. Deformation of the foam rubber core again results in a measurable increase in pressure.

The presented sensor system offers

- high sensitivity in the order of tens of milliNewtons
- flexible size and shape of the sensing elements
- accordingly variable spatial resolution down to around $1\,cm$ square
- inexpensive and rugged design

It is thus intended to complement rather than replace currently employed systems with spatial resolutions in the mm range. These fine sensing devices are optimally suited for the fingers of robotic manipulators where the high resolution pressure "image" delivered by taxel fields supports tasks such as dexterous manipulation. The presented coarse sensor system is suited to cover large parts of the robot's body, where spatial resolution is less of an issue but collisions need to be detected reliably and with high sensitivity.

The contribution of this paper thus lies in demonstrating the viability of a seemingly overlooked inexpensive and robust sensing technology to provide complementary capabilities to those already widely in use.

II. RELATED WORK

A variety of different tactile sensing methodologies have been proposed in the literature. Neither an early survey on robot tactile sensing technology by Nicholls and Lee [3] however, nor the more recent survey on tactile human-robot interaction by Argall and Billard [4], or the Springer Handbook of robotics [5] mention this specific sensor principle.

Among the many different approaches mentioned in the above surveys, we will look at those closely related to the presented method.

One possibility to provide tactile sensing over large areas of the robot body is the use of segments of hard shell attached via several sensing elements, as exemplified in Iwata et al. [6] and Frigola et al. [7]. These hard shells however

have limitations in covering e.g. articulated body parts such as arms.

A similar approach to the one presented here is followed by Syntouch[1]. This fingertip sensor uses a combination of tactile electrodes and a liquid inside a rubber enclosure (instead of air as in our case). It can measure force, vibration and temperature, allowing to estimate tri-axial forces, discriminate various curvatures, textures and object compliance. The sensor is however explicitly designed for fingertips, being available for the Barrett and Shadow hands.

The system employed by Wösch and Feiten [8] used a number of sensor segments in the order of $10 \times 10\,cm$ size, each consisting of a sandwich of two conductive foam layers separated by an insulating net. When a force is applied to the package the conductive foams get into contact, with the resistance at the point of contact depending on the amount of pressure. Electrodes are attached to both ends of the top layer, creating a voltage gradient inside the conductive foam. The voltage at the point of contact thus depends on the position along the gradient. Using two such packages with perpendicular gradients thus allows to detect the point of contact on the surface of the sensor. This system allows sensor segments of various sizes and shapes covering large areas of the robots surface, and is inexpensive and robust. It is however limited to detection of a single contact point.

Minato's et al. [9] CB2 child like robot is covered in a whole-body soft skin consisting of a silicon surface with 197 tactile sensors sandwiched between a urethane form and a silicone skin. The output of these embedded film-type piezoelectric sensors (PVDF) is proportional to the rate of change of bending (deformation rate), and a contact force is obtained by temporal integration of the sensor output. As the deformation of the urethane foam spreads somewhat, deformation can also be measured for forces applied between two sensors. The Robovie series of robots [10], [11] also uses this type of sensor, with up to 276 sensing elements.

Mukai et al. [12] use tiles of piezo-resistive semiconductor pressure sensors to cover the arms and chest of the elastic body of the robot RI-MAN.

The modular skin of pressure-sensitive elements developed by Ohmura et al. [13] can be folded and cut, and coverage modified conveniently by addition/deletion of the modules, which communicate via a serial bus. A sensor consists of a photo-reflector covered by urethane foam, and pressure is detected by the light scattering caused by foam deformation. In [14] the body of a humanoid robot is covered by a skin consisting of 1864 of these sensor elements.

Minato et al. [15] present a pressure sensor consisting of a small hard plastic plate with a spring-like attachment to the robot body, which covers a photo emitter and interrupter pair able to detect changes in pressure. 90 if these sensors are used to cover the body of the humanoid BabyBot.

The system by Yoshikai et al. [16] uses an approach slightly different from the above for the robot *macra*, where the body is studded with 49 3-axis force sensors which are covered by soft urethane foam covering the entire body.

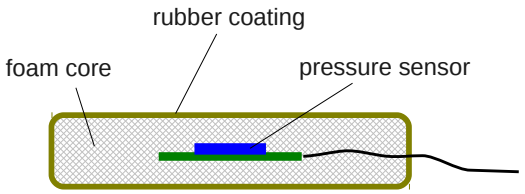


Fig. 1. Design of a sensing element

Weiss Robotics[2] offers a highly sensitive modular tactile sensing system and provides miniaturised elements for equipping e.g. robotic finger tips as well as larger modules (e.g. $24 \times 16\,cm$). But while small elements are also available with crooked measurement plates, as for usage in the fingers of the SCHUNK dexterous hand SDH-2, the rigidity of the larger plates prevents coverage of curved surfaces. Moreover, these pads cost beyond 10 k€ for a complete hand.

Finally, the RoboSkin project[3] [17], [18] aimed at developing sensors for covering large areas of a robot with robotic skin Their sensor system consists of flexible printed circuit boards (PCB), each holding 12 capacitive pressure sensing elements. These PCBs have a triangular shape with a side length of $30\,mm$ and contain the electronics to communicate with their three neighbours, allowing to connect up to 16 $triangles$. Each such assembly is then connected to a micro controller board, which itself is connected to a CAN bus. This arrangement allows for flexible coverage of curved surfaces and minimises the amount of cabling required. The sensors are finally covered by thin layer of silicone rubber resulting in a package that is quite robust.

What distinguishes our approach from most of the above approaches, where many small sensing elements are embedded in a flat soft cushion, is that in our case the cushion itself *is* the sensor, and can cover relatively large and arbitrarily shaped surfaces.

III. APPROACH

The presented sensor system exhibits a simple design, as shown in Figure 1. A sensor pad is comprised of a soft foam core which is covered in an airtight rubber coating. Embedded into the foam core is a barometric pressure sensor. We manufactured several prototypes from foam rubber pieces of various sizes and stiffness and covered them with latex rubber to produce the airtight coating.

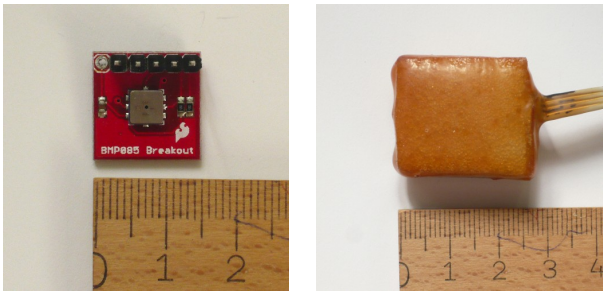
The pressure sensor we used is a Bosch BMP 085[4] (see Figure2(a)). This is a high-precision, ultra-low power digital barometric pressure sensor based on piezo-resistive MEMS technology, selling for around 4 USD in larger quantities. It offers a range of $300 \ldots 1100\,hPa$, with an absolute accuracy of $2.5\,hPa$ and a noise level of down to $0.03\,hPa$ (which

[1] http://www.syntouchllc.com

[2] http://www.weiss-robotics.de

[3] http://www.roboskin.eu

[4] http://www.bosch-sensortec.com/content/language1/html/3477.htm



(a) (b)

Fig. 2. The barometric pressure sensor (left) and a small sensor pad (right)

is equivalent to an altitude change of $0.25\,m$). The sensor is temperature compensated and also provides temperature besides pressure. The sensor itself measures $5 \times 5 \times 1.2\,mm$, however the carrier board we used was slightly larger. The next revision of the sensor, the Bosch BMP180, offers the same characteristics at an even smaller size of $3.65 \times 3.6 \times 0.93\,mm$ and slightly lower price.

The sensor operates at a supply voltage of $1.8 \ldots 3.6\,V$, typically $2.5\,V$, and draws between $3 \ldots 12\,\mu A$. It connects directly to a standard digital two-wire I2C bus for convenient and robust data readout. Some additional electronics is however required as the sensor's I2C address is fixed, such as I2C swtiches like the Texas Instruments TCA9548A or NXP PCA9548A.

The sensor offers 4 sensing modes with different data acquisition times and RMS noise levels: ultra low power mode ($4.5\,ms$, $0.06\,hPa$), standard mode ($7.5\,ms$, $0.05\,hPa$), high resolution mode ($13.5\,ms$, $0.04\,hPa$) and ultra high resolution mode ($25.5\,ms$, $0.03\,hPa$).

The foam core with its rubber coating can have any size and shape in principle. Practically however there are trade offs between sensitivity and size. Moreover the stiffness of the foam core as well as the foam's matrix structure affect sensitivity and can be used to produce a sensitive skin-like sensor vs. a tough bumper.

Compensation of atmospheric pressure variations: Given the high sensitivity of the pressure sensor, it will not only detect applied forces but also atmospheric pressure changes. To compensate for these slow changes in the basic pressure level we implemented an adaptive baseline filter similar to the thermal drift compensation in [18] by maintaining a sliding mean of the baseline pressure (see Algorithm 1). During a short 1 to 2 second calibration phase T after turning on the sensor we store a number of samples in a ring buffer and calculate the baseline mean μ_b and standard deviation σ_b. Then during operation, for a given pressure p we check whether it lies within the σ_b noise band around μ_b, and if yes add it to the ring buffer and update μ_b and σ_b. The function then returns the pressure relative to the baseline. This simple procedure ensures that the sliding mean only takes into account pressures near baseline and is not affected by extended application of an actual force.

Algorithm 1 Adaptive baseline filter

```
function CALIBRATE BASELINE(T)
    while t < T do
        add current pressure p to ring buffer
    end while
    calculate μ_b and σ_b from ring buffer
end function
function ADAPTIVE BASELINE FILTER(p)
    if |p − μ_b| ≤ σ_b then
        add p to ring buffer
        update μ_b and σ_b from ring buffer
    end if
    return p − μ_b.
end function
```

IV. EXPERIMENTAL RESULTS

In our experiments we used the BMP085 sensor in ultra low power mode, operating at $3\,V$ and connected to a PC with a USB-I2C converter operating at $5\,V$ via a logic level shifter. Ultra low power mode is the fasted mode, returning a read out after $4.5\,ms$, as the sensor only takes a single measurement vs. averaging internally over several measurements before returning a read out in the other modes. This comes at the cost of a higher noise level. However we consider speed in reporting a collision more essential than accuracy for this type of sensor.

Experiments were performed by placing defined weights in the centre of the sensor pad, with a contact area of ca. $15mm$ diameter, and waiting until measurement values were stationary.

We tested three different sensor pads: a small one ($2.5 \times 2.5 \times 1\,cm$) shown in Figure 2(b), a medium sized ($4.0 \times 4.0 \times 1.5\,cm$), and a large one ($20.0 \times 14.0 \times 3.5\,cm$), using a soft, medium and rather stiff foam core respectively. Note that due to current limitations of the manufacturing process of the prototypes the large sensor pad was not completely airtight (as can be seen in the pressure curve dipping below 0 in Figure 8 below).

Figures 3 to 5 shows the measured pressure above baseline vs. applied force for each sensor pad. Shown values are averages over 10 measurements with corresponding error bars.

As can be seen the small and medium sensor pads have a roughly linear characteristic, with the softer small pad being more sensitive, as was to be expected. Results for the stiff large sensor pad also indicate a linear characteristic, but due to the fact that this pad was not completely airtight as mentioned above, it was difficult to get reliable sensor readings in this case, as can be seen in the larger error bars. Note the decreasing sensitivity for increasing stiffness of the different foam materials.

Note further that force in the above cases was always applied to the same location on the sensor pad. Depending on the shape of the sensor pad and the distribution of the applied load sensitivity varies across the pad surface, with

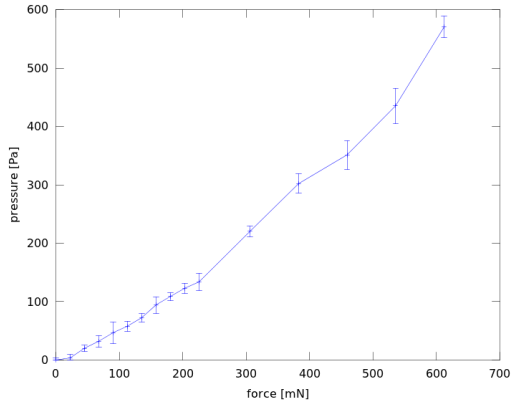


Fig. 3. Measured pressure vs. applied force for the small sensor pad

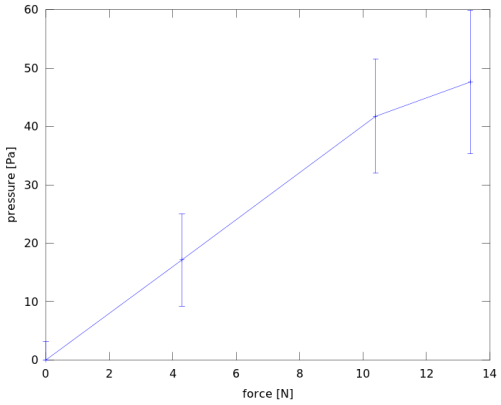


Fig. 5. Measured pressure vs. applied force for the large sensor pad

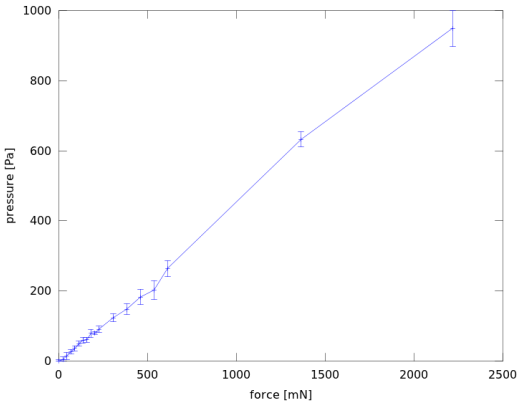


Fig. 4. Measured pressure vs. applied force for the medium sensor pad

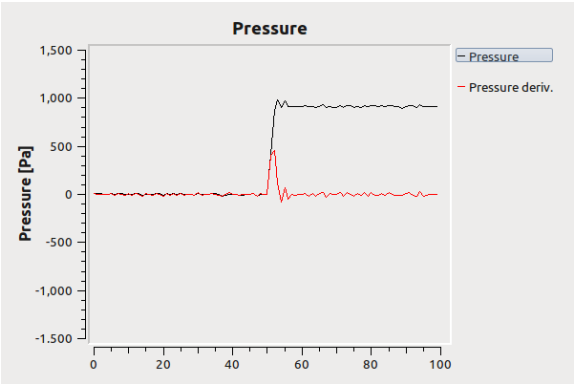


Fig. 6. Dropping a small weight on the medium sensor pad (1 tick represents $50\,ms$)

the centre being most sensitive.

Figures 6 to 8 show qualitative results for the typical dynamic behaviour. Note that one tick on the horizontal axis represents $50\,ms$. Figure 6 shows the pressure and gradient for the medium sensor pad when dropping a small weight of $220\,g$ from a height ca. $1\,cm$. The elasticity of the sensor pad can be clearly seen in the small oscillations. Figure 7 shows the same sensor pad for light tapping with a finger (as in double-clicking on a laptop mouse pad). And Figure 8 shows the large sensor pad when being punched by a fist and hit by a hammer, after which the sensor continued to work normally. Note the different scaling of the vertical axis for each figure.

V. CONCLUSION

We presented a tactile sensing system based on a simple and robust design. The system exhibits characteristics that make it particularly suited for service robotics applications where safe operation in tight spaces requires coverage of large and also curved areas of the robot body, as the sensing pads can be made to have any size and shape. They are robust to wear and also offer mechanical protection by cushioning against impact. The sensor is cheap, highly sensitive (in the order of tens of milliNewtons) and at the same time able to take quite some abuse, such as violent impacts on the sensor pad. Sensors can be daisy-chained on an I2C bus, limiting the amount of required cabling.

On the down side the maximal spatial resolution is an order of magnitude lower than that of systems using arrays of small tactile fields. And while the sensor pads exhibit a more or less linear characteristic, the sensitivity varies across the pad's surface, subject to the mechanical properties of the pad (stiffness of the foam core and the rubber coating).

As such the presented system is intended to complement rather than replace existing technologies, with high resolution sensors covering those robot parts requiring high fidelity tactile feedback for manipulation, and the presented system covering the rest of the robot body.

Future work will concentrate on experimenting with different types of foam rubber (more open vs. more closed pores, integral foams) and coating to maximise sensitivity as well as uniformity of sensitivity across sensor pads. Finally extended trials in harsh conditions (humidity, dust, sunlight, repeated impacts) will be needed to demonstrate the applicability on a wide range of indoor and outdoor robots.

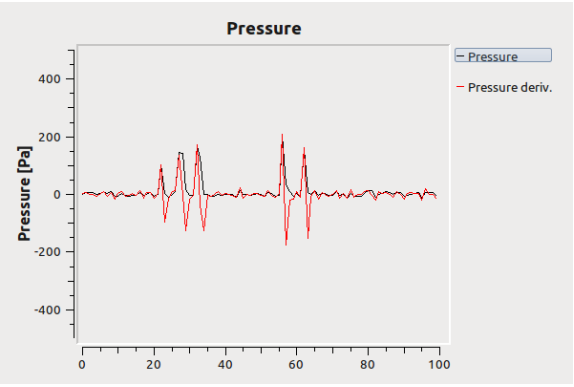


Fig. 7. Tapping the medium sensor pad lightly with a finger (1 tick represents $50\,ms$)

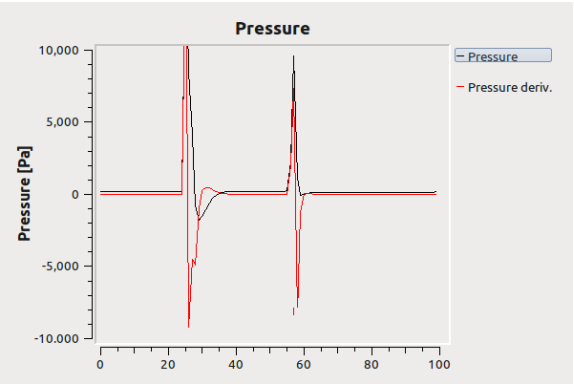


Fig. 8. Hitting the large sensor pad with a fist and hammer. Note that the this particular pad was not completely airtight due to limitations in the manufacturing process, resulting in the pressure temporarily becoming negative w.r.t. to atmospheric pressure. (1 tick represents $50\,ms$)

ACKNOWLEDGEMENT

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement No. 215181, CogX and by the Austrian Science Foundation under grant agreement No. I513-N23.

REFERENCES

[1] Y. Bekiroglu, J. Laaksonen, J. A. Jorgensen, V. Kyrki, and D. Kragic, "Assessing grasp stability based on learning and haptic data," *IEEE Transactions on Robotics*, vol. 27, no. 3, pp. 616–629, 2011.

[2] H. Zhang and N. N. Chen, "Control of contact via tactile sensing," in *IEEE International Conference on Robotics and Automation*, 2000, pp. 482–495.

[3] H. R. Nicholls and M. H. Lee, "A Survey of Robot Tactile Sensing Technology," *The International Journal of Robotics Research*, vol. 8, no. 3, pp. 3–30, 1989.

[4] B. D. Argall and A. G. Billard, "A survey of Tactile HumanRobot Interactions," *Robotics and Autonomous Systems*, vol. 58, no. 10, pp. 1159–1176, 2010.

[5] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, 2008.

[6] H. Iwata, H. Hoshino, T. Morita, and S. Sugano, "Human-humanoid physical interaction realizing force following and task fulfillment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2000.

[7] M. Frigola, A. Casals, and J. Amat, "Humanrobot interaction based on a sensitive bumper skin," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[8] T. Wösch and W. Feiten, "Reactive Motion Control for Human-Robot Tactile Interaction," in *IEEE International Conference on Robotics and Automation*, no. May, 2002, pp. 3807–3812.

[9] T. Minato, Y. Yoshikawa, T. Noda, S. Ikemoto, H. Ishiguro, and M. Asada, "CB2: a child robot with biomimetic body for cognitive developmental robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[10] N. Mitsunaga, T. Miyashita, H. Ishiguro, K. Kogure, and N. Hagita, "Robovie-IV: a communication robot interacting with people daily in an office," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[11] T. Miyashita, T. Tajika, H. Ishiguro, K. Kogure, and N. Hagita, "Haptic communication between humans and robots," *Robotics Research*, vol. 28, pp. 525–536, 2007.

[12] T. Mukai, M. Onishi, T. Odashima, S. Hirano, and Z. Luo, "Development of the tactile sensor system of a human-interactive robot RI-MAN," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 505–512, 2008.

[13] Y. Ohmura, Y. Kuniyoshi, and A. Nagakubo, "Conformable and scalable tactile sensor skin for curved surfaces," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[14] Y. Ohmura and Y. Kuniyoshi, "Humanoid robot which can lift a 30 kg box by whole body contact and tactile feedback," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[15] T. Minato, F. DallaLibera, S. Yokokawa, Y. Nakamura, H. Ishiguro, and E. Menegatti, "A baby robot platform for cognitive developmental robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009.

[16] T. Yoshikai, M. Hayashi, Y. Ishizaka, T. Sagisaka, and M. Inaba, "Behavior integration for whole-body close interactions by a humanoid with soft sensor flesh," in *IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS'07)*, 2007.

[17] G. Cannata, M. Maggiali, G. Metta, and G. Sandini, "An Embedded Artificial Skin for Humanoid Robots," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2008, pp. 2–6.

[18] A. Schmitz, P. Maiolino, M. Maggiali, L. Natale, G. Cannata, and G. Metta, "Methods and Technologies for the Implementation of Large-Scale Robot Tactile Sensors," *IEEE TRANSACTIONS ON ROBOTICS*, vol. 27, no. 3, pp. 389–400, 2011.