Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

An Optimisation-Based Distributed Cooperative Control for Multi-Robot Manipulation with Obstacle Avoidance

Yanhao He * **Min Wu** * **Steven Liu** *

* *University of Kaiserslautern (TUK), Kaiserslautern, Germany*
e-mail: {yanhao, mwu, sliu}@eit.uni-kl.de

Abstract: Multi-robot manipulation systems are usually high-dimensional, kinematically complex and the internal forces are sensitive to robot motion errors due to the physical coupling, especially when the manipulated object is rigid. In this work, a distributed cooperative controller is designed for this scenario. Besides transporting the object, obstacle avoidance and manipulability enhancement are also achieved online by a novel optimisation-based approach. Since the local controller does not require the other robots to send the model or joint-space data, the system is flexible and the communication cost is minimal. Experiments show that no internal force is generated when the robots are changing their poses for the additional tasks and the online computation is fast.

Keywords: Robotics technology, Robot manipulators, Autonomous robotic systems, Cooperative systems, Distributed control and estimation, Multi-agent systems, Industrial applications of optimal control

1. INTRODUCTION

Manipulation and transportation are common tasks for robots in industry and daily service. Due to the complexity and variability of the task details and the environment, multi-robot systems have become quite useful in many situations because they support higher payload and they are more adaptive to changes thanks to their flexibility. Research work on cooperative robot control already exists since the 1990s, such as by Sugar and Kumar (1999, 1998), where the mechanical design and decentralised control of a multi-robot mobile manipulation system are presented. The goal is to let the workpiece track a given trajectory. Erhart et al. (2013) introduced the cooperative control design for two mobile manipulators. In addition to motion error, the internal forces are also recorded and analysed. Similarly a more recent research by He et al. (2018) has focused on internal force suppression in multi-robot transportation. Pure arm systems are also common and of great interest. Salehian et al. (2018) has proposed a unified multi-arm control framework for manipulation tasks, and Behrens et al. (2019) developed a task scheduler and motion planner for a dual-arm manipulator in real industrial application.

Today centralised control is still popular in dual-arm scenarios. For example the aforementioned work of Salehian et al. (2018) includes a centralised inverse kinematic solver and quadratic programming for self-collision avoidance. Another centralisedly controlled dual-arm manipulator is shown by Lin et al. (2018) with the main focus on the robustness against external force disturbances. They use two KUKA LWR arms so 14 joints are controlled online. A responsive coordination motion controller for an ABB YuMi dual-arm system, also with 14 DOFs, has been successfully implemented by Beuke et al. (2018). However for much more complicated systems such as that of Erhart et al. (2013), centralised control would be too computationally demanding.

The challenge with decentralised controller is to guarantee the global coordination. For manipulation this is extremely important because due to the physical coupling via the grasped object, any conflicting motion would generate undesired internal force, as discussed by Erhart and Hirche (2015). Early solutions to this issue, such as by Kume et al. (2002), mostly rely on a leader-follower architecture, however the system scalability is still very limited, and the group performance strongly depends on the leader alone. On the other hand, a distributed and leaderless multi-robot system, e.g. as proposed by Antonelli et al. (2013), often implements an global state observer on each robot, which requires either the detailed model of the other cooperating agents or a lot of communication, making it unsuitable for heterogeneous robot systems in practice.

To reduce internal forces and improve safety in physical interaction with robots, impedance control or PD control with a position reference is often desired for robot arms, however a position-level optimisation problem for redundant manipulators is usually difficult and the solution relies on pure numerical approaches. Programming libraries including NLopt, SciPy, OptimLib and some MATLAB toolboxes do allow users to implement their online solver, and researchers like Ratliff et al. (2009) have developed optimisation algorithms for robotics, but usually the computation time is too long and it takes much effort to tune solver parameters to get a reliable result.

In this paper a novel and fast approach to achieve a position-level optimum is proposed for motion planning

and control in cooperative robotic manipulation. The considered scenario includes unexpected dynamic obstacle, and joint-space constraints are also considered. Based on this solution a coordination strategy is designed so that the robots do not perform conflicting motion and thus no internal force arises even if the manipulated object is rigid. The overall control has a distributed structure with each robot requiring only local sensor data and a limited amount of communication that does not include any joint-space data/model of the other robots. Furthermore, the measurement or estimation of the manipulation forces is not needed. Experiments are done on a dual-arm platform to evaluate the control performance.

2. OPTIMAL MOTION PLANNING

As mentioned above, for robot arms a direct optimisation in the position level is usually hard for online implementation, however a velocity-level optimisation problem can be solved analytically. Therefore the main idea in this part is to firstly obtain an optimal velocity reference, and then transform it into a pose reference.

2.1 Velocity-level solution

Multitask robot control is possible with the null space control law, which originates from an optimisation problem as described by Siciliano et al. (2009), where different priorities must be assigned to the objectives. For example the robot may stick to its desired end effector motion while try to avoid obstacles as a secondary task using its redundancy. But in many cases obstacle avoidance is of the same importance as the main task and it is desired to sacrifice some transportation accuracy in order to be collision-free, so in this work the problem is formulated as

$$\begin{aligned}\min_{\dot{\boldsymbol{q}}_{\mathrm{d0}}}\ g_0 = &\frac{1}{2}\left(\dot{\boldsymbol{x}}_{\mathrm{obj,d0}}-\dot{\boldsymbol{x}}_{\mathrm{trp}}\right)^{\mathrm{T}}\boldsymbol{W}_1\left(\dot{\boldsymbol{x}}_{\mathrm{obj,d0}}-\dot{\boldsymbol{x}}_{\mathrm{trp}}\right)\\ &+\sum_{j=1}^{N_{\mathrm{POI}}}\frac{1}{2}\left(\boldsymbol{v}_{j,\mathrm{d0}}-\boldsymbol{v}_{j,\mathrm{obs}}\right)^{\mathrm{T}}\boldsymbol{W}_{2,j}\left(\boldsymbol{v}_{j,\mathrm{d0}}-\boldsymbol{v}_{j,\mathrm{obs}}\right)\\ &+\frac{1}{2}\left(\dot{\boldsymbol{q}}_{\mathrm{d0}}-\dot{\boldsymbol{q}}_{\mathrm{mpb}}\right)^{\mathrm{T}}\boldsymbol{W}_3\left(\dot{\boldsymbol{q}}_{\mathrm{d0}}-\dot{\boldsymbol{q}}_{\mathrm{mpb}}\right)\end{aligned} \tag{1}$$

where $\dot{\boldsymbol{q}}_{\mathrm{d0}}$ is the desired optimal joint velocity.

The first term in the cost function is related to the main manipulation task. $\dot{\boldsymbol{x}}_{\mathrm{obj,d0}}$ is the object velocity as a consequence of $\dot{\boldsymbol{q}}_{\mathrm{d0}}$, which can be computed using differential kinematics with the assumption that the grasping geometry is known, as

$$\dot{\boldsymbol{x}}_{\mathrm{obj,d0}} = \boldsymbol{J}_{\mathrm{obj}}\dot{\boldsymbol{q}}_{\mathrm{d0}} \tag{2}$$

where $\boldsymbol{J}_{\mathrm{obj}}$ is the Jacobian for the object frame. $\dot{\boldsymbol{x}}_{\mathrm{trp}}$ is the desired object velocity (or "twist") for tracking the transportation trajectory. With known grasping geometry, the Cartesian pose of the object can be calculated using forward kinematics, and the 6-by-1 pose error $\tilde{\boldsymbol{x}}_{\mathrm{obj,trp}}$ relative to the desired transportation pose reference $\boldsymbol{T}_{\mathrm{d,trp}}$ is obtainable. Here it is assumed that $\boldsymbol{T}_{\mathrm{d,trp}}$ is already available to every robot by either a high-level path planner or offline pre-design. $\dot{\boldsymbol{x}}_{\mathrm{trp}}$ can be determined according to $\tilde{\boldsymbol{x}}_{\mathrm{obj,trp}}$ in various ways. In this work a simple P control is chosen as shown in the following Equation (3).

$$\dot{\boldsymbol{x}}_{\mathrm{trp}} = \boldsymbol{K}_{\mathrm{trp}}\tilde{\boldsymbol{x}}_{\mathrm{obj,trp}} \tag{3}$$

where $\boldsymbol{K}_{\mathrm{trp}}$ is the P-gain matrix. $\boldsymbol{W}_1$ is a positive-definite weighting matrix.

The second component of g_0 is for whole-body obstacle avoidance. Firstly several Points of Interest (POIs) that are fixed on the robot body as well as the manipulated object need to be selected. Theoretically all points that have the potential of colliding with an obstacle should be chosen, but that would add too much computational burden to the system, so in practice only those critical ones are needed. For each POI j, its translational velocity $\boldsymbol{v}_{j,\mathrm{d0}}$ can be computed with

$$\boldsymbol{v}_{j,\mathrm{d0}} = \boldsymbol{J}_{v,j}\dot{\boldsymbol{q}}_{\mathrm{d0}} \tag{4}$$

where $\boldsymbol{J}_{v,j}$ is the first three rows of the Jacobian matrix corresponding to the j-th POI.

The robot only needs to detect and react to obstacles that are close enough, and this is achieved by defining a reaction radius r_{obs}. Based on forward kinematics the position of the j-th POI $\boldsymbol{p}_j$ can be obtained online and its distance to the obstacle position $\boldsymbol{p}_{\mathrm{obs}}$ can be computed. The desired avoidance velocity $\boldsymbol{v}_{j,\mathrm{obs}}$ is then computed using the following law (5).

$$\boldsymbol{v}_{j,\mathrm{obs}} = \begin{cases}\boldsymbol{K}_{\mathrm{obs}}\left(\boldsymbol{p}_j-\boldsymbol{p}_{\mathrm{obs}}\right), & \text{if } ||\boldsymbol{p}_j-\boldsymbol{p}_{\mathrm{obs}}|| < r_{\mathrm{obs}}\\ \boldsymbol{0}, & \text{otherwise}\end{cases} \tag{5}$$

where $\boldsymbol{K}_{\mathrm{obs}}$ is again a P control gain. Similarly the weight also has a switching mechanism

$$\boldsymbol{W}_{2,j} = \begin{cases}\boldsymbol{W}_{\mathrm{obs}}, & \text{if } ||\boldsymbol{p}_j-\boldsymbol{p}_{\mathrm{obs}}|| < r_{\mathrm{obs}}\\ \boldsymbol{0}, & \text{otherwise}\end{cases} \tag{6}$$

where $\boldsymbol{W}_{\mathrm{obs}}$ is the positive-definite weighting matrix to use when an obstacle is close to the j-th POI. In practice the obstacle position $\boldsymbol{p}_{\mathrm{obs}}$ can be measured with LiDAR, UWB, infrared, camera sensors, etc. If the sensor does not have high detection accuracy, the reaction radius r_{obs} can be chosen relatively large to ensure safety.

The purpose of the last part of g_0 is to utilise the redundancy of the robot arm to maintain a good manipulability, a criterion indicating the capability of performing further motion or delivering required forces with the current pose. Depending on the use case, this criterion can be defined in various ways. This paper does not discuss its detailed formulation but simply denotes it as $w(\boldsymbol{q})$. $\boldsymbol{W}_3$ is a positive-definite weighting matrix. $\dot{\boldsymbol{q}}_{\mathrm{mpb}}$ is a desired joint velocity for maximising $w(\boldsymbol{q})$, which according to Siciliano et al. (2009) can be computed using

$$\dot{\boldsymbol{q}}_{\mathrm{mpb}} = \boldsymbol{K}_{\mathrm{mpb}}\frac{\partial w\left(\boldsymbol{q}\right)}{\partial \boldsymbol{q}} \tag{7}$$

If $w\left(\boldsymbol{q}\right)$ does not have a closed form, a "best pose" $\boldsymbol{q}_{\mathrm{mpb}}$ can be chosen beforehand either by experience or by an offline optimisation and then the following Equation (8) can be used to determine $\dot{\boldsymbol{q}}_{\mathrm{mpb}}$ in real-time.

$$\dot{\boldsymbol{q}}_{\mathrm{mpb}} = \boldsymbol{K}_{\mathrm{mpb}}\left(\boldsymbol{q}_{\mathrm{mpb}}-\boldsymbol{q}\right) \tag{8}$$

To solve the problem in (1), the position-dependent variables are treated constant, meaning the optimisation is intended only for that instant. Then the quadratic optimisation problem can be solved analytically by letting the gradient of g_0 be zero, which leads to

$$\dot{\boldsymbol{q}}_{\mathrm{d0}} = \boldsymbol{A}^{-1}\boldsymbol{b} \tag{9}$$

where

$$\begin{aligned}\boldsymbol{A} &= \boldsymbol{J}_{\mathrm{obj}}^{\mathrm{T}}\boldsymbol{W}_1\boldsymbol{J}_{\mathrm{obj}} + \sum_{j=1}^{N_{\mathrm{POI}}}\boldsymbol{J}_{v,j}^{\mathrm{T}}\boldsymbol{W}_{2,j}\boldsymbol{J}_{v,j} + \boldsymbol{W}_3 \\ \boldsymbol{b} &= \boldsymbol{J}_{\mathrm{obj}}^{\mathrm{T}}\boldsymbol{W}_1\dot{\boldsymbol{x}}_{\mathrm{trp}} + \sum_{j=1}^{N_{\mathrm{POI}}}\boldsymbol{J}_{v,j}^{\mathrm{T}}\boldsymbol{W}_{2,j}\boldsymbol{v}_{j,\mathrm{obs}} + \boldsymbol{W}_3\dot{\boldsymbol{q}}_{\mathrm{mpb}}\end{aligned} \tag{10}$$

The joint-space optimisation leads to the existence of the positive-definite weight $\boldsymbol{W}_3$ in $\boldsymbol{A}$, making $\boldsymbol{A}$ always invertible and therefore the solution (10) is robust against singularities.

2.2 Position-level solution

The $\dot{\boldsymbol{q}}_{\mathrm{d0}}$ obtained in the previous section is not compatible with many position-based motion controllers. Hence an iterative process is proposed here to convert it into a position-level reference.

The algorithm first multiplies $\dot{\boldsymbol{q}}_{\mathrm{d0}}$ by a small enough time step Δt_{itr} to obtain a position-level increment, and starting from the actual pose $\boldsymbol{q}_{\mathrm{now}}$, a pose reference $\boldsymbol{q}_{\mathrm{d0}}$ will be updated and then fed into the next iteration, where a new $\dot{\boldsymbol{q}}_{\mathrm{d0}}$ will be computed using Equation (10), and the incrementation repeats until a specified time horizon T_{itr} is reached or $\dot{\boldsymbol{q}}_{\mathrm{d0}}$ settles at almost zero.

The basic principle is the same as some Jacobian-based inverse kinematics algorithms as introduced by Siciliano and Khatib (2008) and Siciliano et al. (2009). The idea behind this is to run a small simulation inside the motion planning algorithm with the robot dynamics modelled as a pure integrator. By taking the actual states as the initial condition and looking into a future horizon, the steady-state robot pose under the optimal velocity control law will be predicted. This final pose is considered a position-level optimum and provided to the robot motion/torque controller as a reference. Since in each iteration $\dot{\boldsymbol{q}}_{\mathrm{d0}}$ is solved analytically using Equation (10), the computation is fast enough for online responsive control. In fact computation times at millisecond level have been observed in experiments. To summarise, a pseudocode is attached as Algorithm 1.

Algorithm 1 Position-level optimisation

```
function LocalOptim(q_now, x_d,trp, p_obs)
    Initialise q_d0 = q_now, t_itr = 0
    repeat
        Compute the object pose based on q_d0
        Compute POI positions based on q_d0
        Compute Jacobians based on q_d0
        Compute ẋ_trp, v_j,obs's, W_2,j's, q̇_mpb
        q̇_d0 = A^{-1} b
        q_d0 = q_d0 + q̇_d0 Δt_itr
        t_itr = t_itr + Δt_itr
    until t_itr ≥ T_itr                ▷ or until q̇_d0 & q̈_d0 ≈ 0
    return q_d0
end function
```

3. COORDINATION STRATEGY

The motion planner presented in the previous chapter is only for one robot locally. If in a multi-robot manipulation system each robot moves according to its own optimum, conflicting motion will happen. In order to achieve coordination, all robot members must have an agreement on the desired object motion and strictly stick to it, even though this may indicate a compromise and lead to some sacrifice of local optimality. Based on this thought, communication is added to the system to allow robots to negotiate.

First the robot would generate the locally optimal motion reference $\boldsymbol{q}_{\mathrm{d0}}$ using Algorithm 1, however this value will not be sent to the controller. Instead, its corresponding object pose $\boldsymbol{T}_{\mathrm{obj,d0}}$ will then be calculated according to the forward kinematic model and broadcast to the whole group. An average pose of all local optimal poses is computed to serve as the negotiation result, denoted as $\boldsymbol{T}_{\mathrm{obj,d}}$, and it must be fulfilled by every robot member.

Since the motion task is changed after negotiation a new motion planning problem is formulated. Now the optimisation problem is constrained by the global agreement $\boldsymbol{T}_{\mathrm{obj,d}}$:

$$\begin{aligned}\min_{\dot{\boldsymbol{q}}_{\mathrm{d}}}\ g = &\sum_{j=1}^{N_{\mathrm{POI}}}\frac{1}{2}\left(\boldsymbol{v}_{j,\mathrm{d}}-\boldsymbol{v}_{j,\mathrm{obs}}\right)^{\mathrm{T}}\boldsymbol{W}_{2,j}\left(\boldsymbol{v}_{j,\mathrm{d}}-\boldsymbol{v}_{j,\mathrm{obs}}\right) \\ &+\frac{1}{2}\left(\dot{\boldsymbol{q}}_{\mathrm{d}}-\dot{\boldsymbol{q}}_{\mathrm{mpb}}\right)^{\mathrm{T}}\boldsymbol{W}_3\left(\dot{\boldsymbol{q}}_{\mathrm{d}}-\dot{\boldsymbol{q}}_{\mathrm{mpb}}\right) \\ \text{s.t.}\quad &\boldsymbol{J}\left(\boldsymbol{q}\right)\dot{\boldsymbol{q}}_{\mathrm{d}} = \dot{\boldsymbol{x}}_{\mathrm{obj,d}}\end{aligned} \tag{11}$$

where $\dot{\boldsymbol{x}}_{\mathrm{obj,d}}$ is a desired Cartesian velocity for tracking the global optimum $\boldsymbol{T}_{\mathrm{obj,d}}$, computed in a similar way to $\dot{\boldsymbol{x}}_{\mathrm{trp}}$ as in Equation (3).

Compared with g_0 in (1), the removal of the transportation component is noticeable. This is because the motion of the object is already constrained. For the same reason any obstacle avoidance POI on the grasped object should also be neglected. The other velocity variables and weights remain the same, but since it is now a multi-robot situation, it is worth highlighting that all vectors are transformed into the robot local coordinate frame. The resulting $\dot{\boldsymbol{q}}_{\mathrm{d}}$ will be the final optimal joint velocity with global consideration.

The method of Lagrangian multipliers are used to solve this equality-constrained optimisation problem, leading to

$$\dot{\boldsymbol{q}}_{\mathrm{d}} = \boldsymbol{\Gamma}^{-1}\left(\boldsymbol{\Theta} + \boldsymbol{J}_{\mathrm{obj}}^{\mathrm{T}}\boldsymbol{\lambda}\right) \tag{12}$$

with

$$\begin{aligned}\boldsymbol{\Gamma} &= \sum_{j=1}^{N_{\mathrm{POI}}}\boldsymbol{J}_{v,j}^{\mathrm{T}}\boldsymbol{W}_{2,j}\boldsymbol{J}_{v,j} + \boldsymbol{W}_3 \\ \boldsymbol{\Theta} &= \sum_{j=1}^{N_{\mathrm{POI}}}\boldsymbol{J}_{v,j}^{\mathrm{T}}\boldsymbol{W}_{2,j}\boldsymbol{v}_{j,\mathrm{obs}} + \boldsymbol{W}_3\dot{\boldsymbol{q}}_{\mathrm{d}}\end{aligned} \tag{13}$$

and

$$\boldsymbol{\lambda} = \left(\boldsymbol{J}_{\mathrm{obj}}\boldsymbol{\Gamma}^{-1}\boldsymbol{J}_{\mathrm{obj}}^{\mathrm{T}}\right)^{-1}\left(\dot{\boldsymbol{x}}_{\mathrm{obj,d}} - \boldsymbol{J}_{\mathrm{obj}}\boldsymbol{\Gamma}^{-1}\boldsymbol{\Theta}\right) \tag{14}$$

Equation (12-14) form the final solution of this second optimisation problem. Note that due to Equation (14), this solution is no longer robust against singularity. A simple workaround in practice is to use the damped least squares inverse to avoid dangers.

With this new analytical solution, the iterative process introduced in Section 2.2 is used again to compute a position-level optimum, which will be fed into the low-level robot controller. The global synchronisation of the object motion guarantees that the grasping formation is maintained. No internal force will thus be produced since

there is no tendency of deformation. Additionally, when one robot has to avoid an obstacle, the intended avoidance motion will be propagated to the other members via negotiation, resulting in a group cooperation for collision avoidance.

The whole design can be intuitively explained as follows: A robot first plans a motion based on its own demand, and then shares its proposal with the other members. After negotiation a compromised solution will be decided, and the robot searches again for the locally best behaviour under this agreement. It can be seen that from a local view the use of multiple robots has introduced new physical constraints. For example, compared with the un-coordinated case, smaller motion for obstacle avoidance would probably be observed with coordination, because the locally optimal motion might push the other robots towards ill poses and they would act against this tendency. By synthesising all the optima, these additional constraints are implicitly tackled in the optimisation problem (11). The pseudocode of the coordinated design is attached as Algorithm 2.

Algorithm 2 Coordinated optimisation

function GROUPOPTIM($\boldsymbol{q}_{\text{now}}$, $\boldsymbol{x}_{\text{d,trp}}$, $\boldsymbol{p}_{\text{obs}}$)
  $\boldsymbol{q}_{\text{d0}} = \text{LocalOptim}(\boldsymbol{q}_{\text{now}}, \boldsymbol{x}_{\text{d,trp}}, \boldsymbol{p}_{\text{obs}})$
  $\boldsymbol{T}_{\text{obj,d0}} = \boldsymbol{f}(\boldsymbol{q}_{\text{d0}})$
  Broadcast $\boldsymbol{T}_{\text{obj,d0}}$
  $\boldsymbol{T}_{\text{obj,d}} = \text{mean}(\text{all } \boldsymbol{T}_{\text{obj,d0}}\text{'s})$
  Initialise $\boldsymbol{q}_{\text{d}} = \boldsymbol{q}_{\text{now}}$, $t_{\text{itr}} = 0$
  repeat
    Compute the object pose based on $\boldsymbol{q}_{\text{d}}$
    Compute POI positions based on $\boldsymbol{q}_{\text{d}}$
    Compute Jacobians based on $\boldsymbol{q}_{\text{d}}$
    Compute $\boldsymbol{v}_{j,\text{obs}}$'s, $\boldsymbol{W}_{2,j}$'s, $\dot{\boldsymbol{q}}_{\text{mpb}}$
    $\boldsymbol{\lambda} = \left(\boldsymbol{J}_{\text{obj}}\boldsymbol{\Gamma}^{-1}\boldsymbol{J}_{\text{obj}}^{\text{T}}\right)^{-1}\left(\dot{\boldsymbol{x}}_{\text{obj,d}} - \boldsymbol{J}_{\text{obj}}\boldsymbol{\Gamma}^{-1}\boldsymbol{\Theta}\right)$
    $\dot{\boldsymbol{q}}_{\text{d}} = \boldsymbol{\Gamma}^{-1}\left(\boldsymbol{\Theta} + \boldsymbol{J}_{\text{obj}}^{\text{T}}\boldsymbol{\lambda}\right)$
    $\boldsymbol{q}_{\text{d}} = \boldsymbol{q}_{\text{d}} + \dot{\boldsymbol{q}}_{\text{d}}\Delta t_{\text{itr}}$
    $t_{\text{itr}} = t_{\text{itr}} + \Delta t_{\text{itr}}$
  until $t_{\text{itr}} \geq T_{\text{itr}}$ ▷ or until $\dot{\boldsymbol{q}}_{\text{d}}\&\ddot{\boldsymbol{q}}_{\text{d}} \approx 0$
  return $\boldsymbol{q}_{\text{d}}$
end function

An advantage of this approach is that the robot does not have to sense the whole environment or know the whole system model, because any run-time demand of the other members will be included in the shared task-space information, and this also makes it easier to expand the multi-robot system with different types of manipulators. In this motion planning stage, only kinematic data are used, so there is no need to measure the manipulation forces online, which is normally expensive. Another main strength is the minimal amount of transferred data. The only message to be sent is a pose variable, normally costing less than 100 bytes.

4. ROBOT ARM CONTROL

As mentioned before, for better safety and controllability, impedance control is often preferred in physical interaction with robot manipulators. The desired behaviour of impedance control in Cartesian space is the same as a mass-spring-damper with totally customisable dynamic parameters, but its implementation is often difficult, therefore in reality simpler alternatives are often used. If the exact decoupled Cartesian space behaviour is not necessary and the robot will mainly perform slow to medium motion, a joint-space PD control can be considered. Its torque control law

$$\boldsymbol{\tau}_{\text{cmd}} = \boldsymbol{K}_{\text{P}}\tilde{\boldsymbol{x}} + \boldsymbol{K}_{\text{D}}\dot{\tilde{\boldsymbol{x}}} \tag{15}$$

is actually implemented in this work, with $\boldsymbol{K}_{\text{P}}$ and $\boldsymbol{K}_{\text{D}}$ being the stiffness and damping coefficients. $\tilde{\boldsymbol{q}}$ and $\dot{\tilde{\boldsymbol{q}}}$ the joint pose and velocity error, respectively.

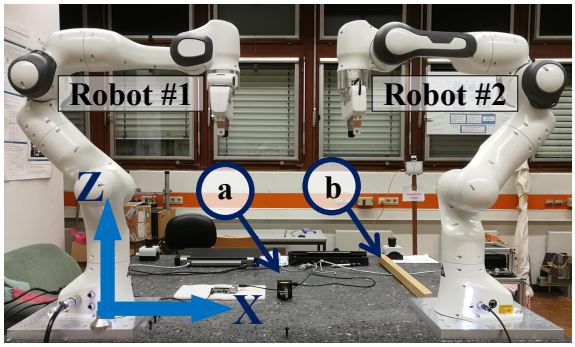

Fig. 1. The experimental setup in the robotics lab of LRS, where (a) is a 2D laser scanner and (b) is a wooden stick as an obstacle

5. EXPERIMENT

5.1 Set-up

To evaluate the control performance, experiments are conducted in the robotics laboratory of the Institute of Control Systems (LRS) in the University of Kaiserslautern. The dual-arm platform shown in Fig. 1 consists of two Franka Emika Panda 7-DOF robot arms. The control algorithm is implemented in C++ on two Ubuntu laptops with the PREEMPT_RT patch.

As shown in Fig. 1, Robot 1 is installed at the origin of the world coordinate frame, while Robot 2 is on the other side, opposing it. A Hokuyo laser scanner is used for obstacle detection. Due to its 2D nature, the robot control and obstacle detection have to be restricted in the XZ plane, and a wooden stick is used as the dynamic obstacle. The detection algorithm is implemented on a Raspberry Pi 3B, which is in the same local area network as the two robots. To simplify the communication topology, this Raspberry Pi also serves as a data exchanging hub for the negotiation, and it is also responsible for the computation of $\boldsymbol{T}_{\text{obj,d}}$. These adjustments however do not change the control structure.

The location of Robot 2 is (0.956, 0, 0). For both robots the initial joint pose is $[0, -\pi/4, 0, -3\pi/4, 0, \pi/2, \pi/4]^{\text{T}}$. This initial pose, as illustrated in Fig. 1, is used in all the official programming examples by Franka and therefore in the following experiments it is assumed to have the best manipulability, i.e. these values are assigned to $\boldsymbol{q}_{\text{mpb}}$ and Equation (8) is used. In all tests, the transportation motion reference is a constant pose that coincides the initial tense-free pose of the grasped object. The parameters are chosen as follows: $\boldsymbol{K}_{\text{trp}}$=3$\boldsymbol{I}$, $\boldsymbol{W}_1$=$\boldsymbol{I}$, $\boldsymbol{K}_{\text{obs}}$=2$\boldsymbol{I}$, $\boldsymbol{W}_{\text{obs}}$=30$\boldsymbol{I}$, r_{obs}=0.3m, $\boldsymbol{K}_{\text{mpb}}$=$\boldsymbol{I}$, $\boldsymbol{W}_3$=0.2$\boldsymbol{I}$. For obstacle avoidance

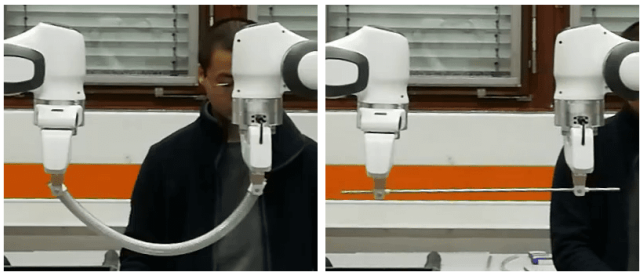

Fig. 2. The initial grasping states of the two objects, the elastic tube on the left and the steel stud on the right

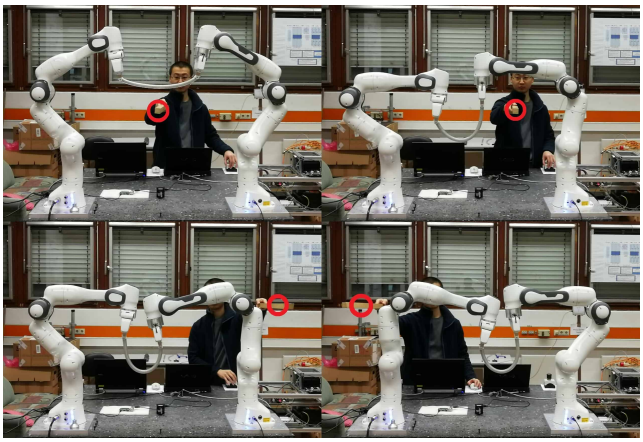

Fig. 3. The robot poses during the experiment without coordination, the object being elastic

the selected POIs are the object centre, the gripper, the wrist and elbow joints of the robot. A fixed time horizon T_{itr}=5s and time step Δt_{itr}=0.01s are set for the iterative process, which means it always performs 500 iterations.

In order to observe the effect of internal forces, an elastic tube and a rigid steel stud will be used as the manipulated objects in different experiments, as shown in Fig. 2. The same test routine is designed for the manipulation of both workpieces, and the controllers with and without coordination are both tested. A video of these experiments can be found on YouTube: https://youtu.be/4EPGzOdIQds

5.2 Results

First the elastic tube is manipulated. During the test the human operator moved the wooden stick to four different positions to see how the robots react. The resulting robot configurations without the negotiation mechanism can be seen in Fig. 3. The detected obstacle position is marked with a red circle. As a consequence of the conflicting motion caused by obstacle avoidance the tube has experienced very obvious deformation in comparison with Fig. 2. The situation with coordinated motion planning is shown in Fig. 4, where the robots successfully maintain the grasping geometry while avoiding the obstacle. The shape of the elastic object is almost unchanged.

Then with the steel stud the same routine is repeated. Because of the rigidity, no matter with or without negotiation the object has not visibly deformed during obstacle avoidance, so the photos are not attached. But now the grasp geometry is known and fixed, therefore the internal

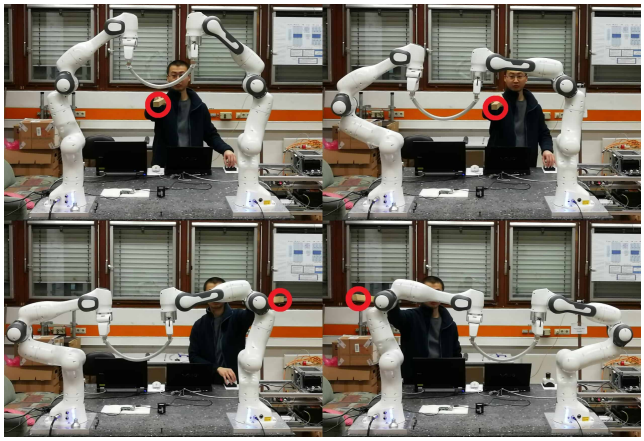

Fig. 4. The robot poses during the experiment with coordination, the object being elastic

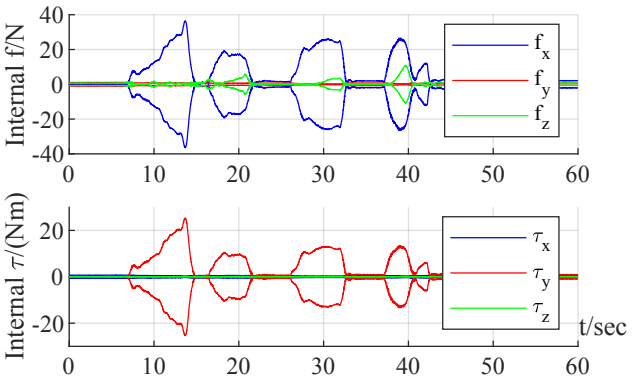

Fig. 5. The internal forces on a rigid object without coordinated motion planning

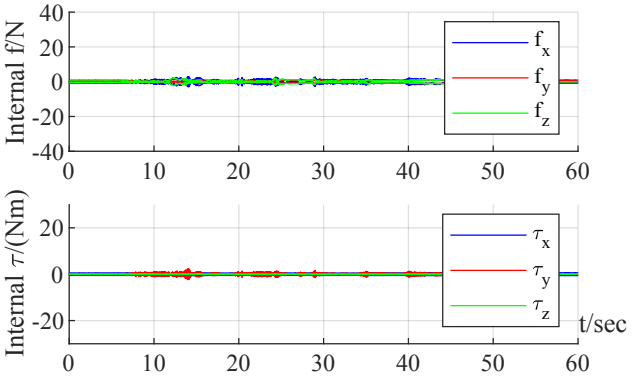

Fig. 6. The internal forces on a rigid object with coordinated motion planning

forces can be analysed using the virtual linkage model proposed by Williams and Khatib (1993). Fig. 5 shows the result without negotiation. A 40N conflicting force along X in the object frame and 20Nm internal torque about Y can be observed. This is large enough to damage some fragile materials. In Fig. 6, where the coordination strategy is applied to get a synchronised object motion reference, hardly any internal stress can be observed.

The costs during the one-minute experiment without coordination are shown in Fig. 7. Since the optimisation cost functions are formulated for the velocity level, the displayed cost here is the integration of g_0 over the 500

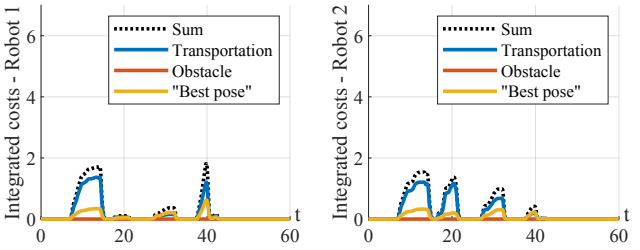

Fig. 7. The integrated costs in the test with a rigid object and without coordination

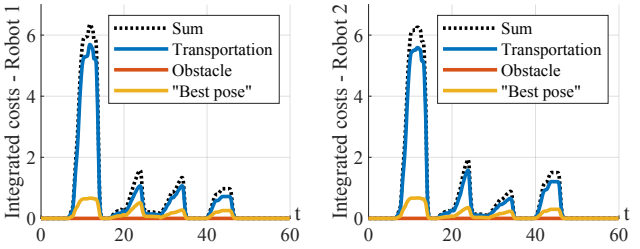

Fig. 8. The integrated costs in the test with a rigid object and with coordination

iterations in each program cycle. The values of different cost components are also plotted in the figures. When the human operator moves the wooden stick close to the robots, an obvious rise appears on the curve. Since the weights for obstacle avoidance are large, the robots react fast enough and this cost component always diminishes immediately so the red curve experiences almost no change.

In Fig. 8 the costs with active coordination are shown, and they are obviously larger than in the no-coordination case. The reason is that g_0 is designed only to indicate the local performance. As discussed in Section 3, after negotiation with the other members, the robot has to compromise and sacrifice part of its original optimality in order to achieve synchronisation and thus avoid dangerous internal load, so the increase of the local costs is actually justifiable.

On one laptop with Intel Core i7-3520M and 8GB RAM, the average online computation time is 3 milliseconds without coordination, and 6 milliseconds otherwise. On the other slower laptop with Intel Core 2 Duo CPU P8700 and 4GB RAM, the computation time cost is 13 milliseconds. Since T_{itr}=5s is actually a conservative choice, by shortening the horizon the computation could be even faster.

6. CONCLUSION

This paper presents a novel approach for fast optimal motion planning. An optimal velocity reference is firstly analytically obtained and then through an iterative process an optimal pose can be calculated and fed into impedance control. The online computation can be finished within a few milliseconds on a modern computer, therefore it is very suitable for real implementation. Based on this method, a coordination strategy is designed for multi-robot manipulation tasks to avoid conflict in unexpected situations, e.g. during obstacle avoidance. Experiments demonstrate that the proposed control strategy can successfully suppress internal forces with minimal communication effort and only local sensing.

REFERENCES

Antonelli, G., Arrichiello, F., Caccavale, F., and Marino, A. (2013). Decentralized centroid and formation control for multi-robot systems. In *2013 IEEE International Conference on Robotics and Automation*. IEEE.

Behrens, J.K., Lange, R., and Mansouri, M. (2019). A constraint programming approach to simultaneous task allocation and motion scheduling for industrial dual-arm manipulation tasks. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE.

Beuke, F., Alatartsev, S., Jessen, S., and Verl, A. (2018). Responsive and reactive dual-arm robot coordination. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Erhart, S. and Hirche, S. (2015). Internal force analysis and load distribution for cooperative multi-robot manipulation. *IEEE Transactions on Robotics*, 31(5), 1238–1243.

Erhart, S., Sieber, D., and Hirche, S. (2013). An impedance-based control architecture for multi-robot cooperative dual-arm mobile manipulation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE.

He, Y., Wu, M., and Liu, S. (2018). Decentralised cooperative mobile manipulation with adaptive control parameters. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE.

Kume, Y., Hirata, Y., Wang, Z.D., and Kosuge, K. (2002). Decentralized control of multiple mobile manipulators handling a single object in coordination. In *IEEE/RSJ International Conference on Intelligent Robots and System*. IEEE.

Lin, H.C., Smith, J., Babarahmati, K.K., Dehio, N., and Mistry, M. (2018). A projected inverse dynamics approach for multi-arm cartesian impedance control. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Ratliff, N., Zucker, M., Bagnell, J.A., and Srinivasa, S. (2009). CHOMP: Gradient optimization techniques for efficient motion planning. In *2009 IEEE International Conference on Robotics and Automation*. IEEE.

Salehian, S.S.M., Figueroa, N., and Billard, A. (2018). A unified framework for coordinated multi-arm motion planning. *The International Journal of Robotics Research*, 37(10), 1205–1232.

Siciliano, B. and Khatib, O. (eds.) (2008). *Springer Handbook of Robotics*. Springer Berlin Heidelberg.

Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2009). *Robotics*. Springer London.

Sugar, T. and Kumar, V. (1998). Decentralized control of cooperating mobile manipulators. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*. IEEE.

Sugar, T. and Kumar, V. (1999). Multiple cooperating mobile manipulators. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*. IEEE.

Williams, D. and Khatib, O. (1993). The virtual linkage: a model for internal forces in multi-grasp manipulation. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press.