Combined Workspace Monitoring and Collision Avoidance for Mobile Manipulators

Angelika Zube
Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB, Karlsruhe, Germany
Email: angelika.zube@iosb.fraunhofer.de

***Abstract*—For safe human-robot interaction and co-existence, collision avoidance is a fundamental prerequisite. Therefore, in this contribution a Nonlinear Model Predictive Control approach for fixed-base and mobile manipulators is presented that allows for avoiding self-collisions and collisions with static and dynamic obstacles while performing tasks defined in the Cartesian space. The collision avoidance takes not only the end-effector but the complete robot consisting of both platform and manipulator into account and relies on a 3D obstacle representation obtained by fusing information from multiple depth sensors. The obstacle representation is applicable to all kinds of objects. It considers occlusions behind the obstacles and the robot to make a conservative assumption on the obstacle size. In order to achieve realtime reactions to obstacles, the obstacle information used in one control step is restricted to the most relevant obstacles determined by distance computation. The method is validated by means of simulation and by application to an omnidirectional mobile manipulator with 10 degrees of freedom.**

I. Introduction

In contrast to current factory automation where humans and robots are strictly separated, many applications could benefit from shared human-robot workspaces. For example, mobile manipulators can supply different work stations with parts and perform standard assembly tasks, while human workers perform more complex tasks in the same workspace. To allow for such shared human-robot workspaces in cluttered environments, robots have to be able to avoid collisions with static and dynamic obstacles while they are executing their original tasks. This involves both the monitoring of the robot environment to detect obstacles and the motion control that has to be able to avoid collisions while moving the robot along reference trajectories determined in a high level planning layer in order to fulfill the robot task.

Especially in large workspaces of mobile manipulators, detecting obstacles near the robot is still challenging. All relevant space around the robot has to be covered and the monitoring has to be carried out in 3D. Due to safety needs, occlusions behind the robot and behind obstacles have to be considered. This is particularly important, when an object is located between the robot and the observing sensor. To reduce these occlusions and to enlarge the observed space, information from multiple sensors has to be fused.

Furthermore, the robot has to react adequately to the detected obstacles to avoid collisions and guarantee safety. An obvious reaction to an upcoming collision is to stop the robot. But this impedes an efficient task execution. Therefore, control strategies are necessary that take current obstacle detections into account to avoid collisions by evasive motions while continuing to move the robot according to its task.

978-1-4673-7929-8/15/$31.00

In this contribution, a Nonlinear Model Predictive Controller that controls the end-effector pose of redundant fixed-base and mobile manipulators according to Cartesian reference trajectories is enhanced and combined with a 3D workspace monitoring approach. As the controller input is a Cartesian trajectory, the robot redundancy can be used for evasive motions to avoid collisions. The obstacles in the robot environment are detected online so that also dynamic obstacles can be handled. Both monitoring and control are designed in order to meet the realtime requirements.

A. Related Work

With regard to shared human-robot workspaces, an active collision avoidance strategy is presented in [1]. The distance between a robot manipulator and humans in its environment is monitored. With decreasing distance the robot is slowed down and then stopped, interrupting the task execution. If the distance continues to decrease, the robot moves the end-effector away from the human.

A common approach for collision avoidance without task interruption is the artificial potential field method introduced by Khatib [2]. The robot task is described by attractive forces acting on the end-effector. Obstacles are transformed into repulsive forces pushing the robot away from the obstacles. These repulsive forces are converted into joint torques, which are projected into the nullspace of the robot. The potential field approach is applied to mobile manipulators in [3] and [4]. In [3], collision avoidance is achieved for static obstacles considering the nonpoint geometry of the robot. To overcome the issue that the computation of potential fields is in general very time consuming [5], in [4] dynamic obstacles are considered in a close range around the end-effector.

Similar to potential fields, a repulsive motion is computed based on kinetostatic safety fields in [6]. The kinetostatic safety field fuses information on the position, velocity, shape and size of the possibly colliding objects. In order to generate the kinetostatic safety field, a human model consisting of boxes is fitted in the obstacle measurements of a depth camera. Collision-free joint motions are obtained by nullspace projection.

In [7], a control algorithm for collision avoidance is proposed that distinguishes between possible collisions with the end-effector and with the robot body. Collisions with the end-effector are avoided similar to the potential field method. To avoid collisions with the robot body, the allowed joint velocities are restricted depending on the minimum distances between points on the robot and obstacles. The distance computation is based on a single depth camera and is suitable for fixed-base manipulators.

In [8], a collision avoidance method for a manipulator mounted on a remote controlled mobile base is presented. For detecting possible collisions, obstacle measurements and manipulator points are projected onto a plane in front of the robot. The mobile base is assumed to be moved on a collision-free path by the teleoperator.

B. Approach

In contrast to the previously described work, this paper presents a collision avoidance strategy based on Nonlinear Model Predictive Control (NMPC) applicable to fixed-base and mobile manipulators. As shown in [9], NMPC is a suitable approach to control the end-effector pose of redundant robots in Cartesian space. Compared to approaches based on nullspace projection like artificial potential field methods, NMPC has the advantage that both an objective function and hard constraints (e.g., joint limits) may be directly considered. Here, this control approach is extended and combined with a 3D workspace monitoring method [10], in order to avoid self-collisions of the robot and collisions with static and dynamic obstacles in parallel to the Cartesian task execution.

The obstacles and their occlusions are represented in a 3D octree structure. The obstacles with closest distances to the robot are regarded as the most relevant obstacles. Using in each control step only these relevant obstacles for avoiding collisions with all robot parts allows for fast reactions despite the 3D representation incorporating obstacle and occlusion information from multiple depth sensors.

Due to the prediction, the algorithm generates a foresightful robot behavior that is especially useful for dynamic obstacles. Not only the current relation of robot and obstacles is used, but also future robot configurations are incorporated. So automatically a different behavior between approaching an obstacle and moving away from obstacles is achieved.

This paper is organized as follows: In Section II and Section III, the underlying workspace monitoring and the Nonlinear Model Predictive Cartesian Control approach are summarized. Extensions in order to avoid self-collisions and to avoid collisions with static and dynamic obstacles based on the 3D obstacle representation are presented in Section IV. Collision avoidance results obtained by means of simulation and on a real 10 DoF mobile manipulator are shown in Section V. Finally, conclusions are presented in Section VI.

II. Workspace Monitoring

In order to allow for avoiding collisions in the presence of humans or other obstacles, sensor-based monitoring of static and dynamic obstacles in the robot's environment is necessary. The requirements for workspace monitoring described in Section I are met by the obstacle representation method presented in [10]. The method is summarized in the following for better comprehensiveness of the paper.

The information about obstacles in the robot workspace is obtained from multiple depth sensors (e.g., laser scanners, Kinect sensors, depth cameras) mounted on the robot or in the robot cell. First, the measured depth data of the individual depth sensors is pre-processed. The sensor data is filtered to distinguish between measurements belonging to the robot and measurements of objects in the robot workspace. For this purpose, the measurements are compared to the geometric robot model using the *Realtime URDF Filter* [11]. Then the data is transferred to point clouds and transformed to a common coordinate frame, so that the following processing steps become sensor independent. Additionally, the space that is in the field of view of each sensor is computed.

Based on the pre-processed sensor data, the space that is occupied or occluded by obstacle measurements and the space that is occluded by the robot is computed by means of ray tracing using an octree data structure [12]. The free space of a sensor is then the space in the field of view of the sensor without the space occupied or occluded by obstacles or by the robot.

Finally, the information of all sensors is fused. All space that is occupied or occluded by an obstacle and cannot be guaranteed to be free by another sensor is interpreted as obstacle. That means, for safety reasons the approach makes a conservative assumption on the obstacle size, which is illustrated in Fig. 1. In order to reduce the computational load, the obstacle representation is restricted to the robot's close range that can be reached by at least one part of the robot within a certain time span. The obstacle representation is updated when new measurements are available so that it can cope with dynamic obstacles as well.

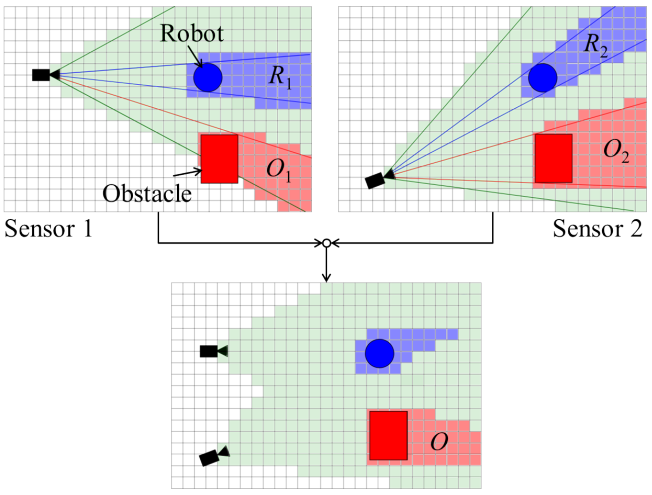


Fig. 1. Workspace monitoring principle with two sensors: fusion of information about occupied and occluded space results in the final obstacle representation O.

III. Cartesian Control

For redundant fixed-base or mobile manipulators in shared human-robot workspaces, control algorithms are necessary that allow the robot to perform a task defined in the Cartesian space and that simultaneously realize additionally desired robot behaviors like avoiding collisions with humans or other obstacles. Therefore, a Nonlinear Model Predictive Control (NMPC) approach has been developed to move the end-effector of a redundant robot along a reference trajectory $\boldsymbol{y}_\mathrm{t}(t)$ containing the desired 3D end-effector position $\boldsymbol{y}_{\mathrm{t,p}}(t)$ and the orientation $\boldsymbol{y}_{\mathrm{t,o}}(t)$. Due to the underlying general robot model, the control algorithm is applicable to both fixed and mobile manipulators. The control algorithm has been presented in [9] and is summarized in the following.

A. Kinematic Robot Model

The (mobile) manipulator is modeled as a chain of rigid links connected by n revolute or prismatic joints. The general kinematic model disregarding accelerations is then given by

$$\dot{\boldsymbol{q}} = \boldsymbol{v} \tag{1}$$

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{q}) \tag{2}$$

with the revolute or prismatic joint positions $\boldsymbol{q} \in \mathbb{R}^n$, the joint velocities $\boldsymbol{v} \in \mathbb{R}^n$ and the end-effector pose $\boldsymbol{y} \in \mathbb{R}^m$. The end-effector pose consists of the 3D position $\boldsymbol{y}_\mathrm{p}$ and the orientation $\boldsymbol{y}_\mathrm{o}$ represented by a unit quaternion ($m = 7$). The direct kinematics function $\boldsymbol{f}(\boldsymbol{q})$ maps the joint configuration $\boldsymbol{q}$ to the Cartesian end-effector pose $\boldsymbol{y}$.

The direct kinematics are computed by assigning a coordinate frame to each joint according to the Denavit-Hartenberg convention and chaining the coordinate transforms of all joints [13]. This is a common procedure for fixed-base manipulators. In order to extend it to mobile manipulators, the mobile platform is described by two prismatic joints for the translational motion and a revolute joint for the rotation about the vertical axis. In the case of non-holomic platforms, additional non-holomic constraints are defined, that can be handled by the presented Nonlinear Model Predictive Controller as well. Hence, a generic kinematic robot model is available that can be applied to both fixed-base and mobile manipulators.

B. Cartesian Nonlinear Model Predictive Control

Based on the generic kinematic robot model presented in the previous Section, a Nonlinear Model Predictive Control (NMPC) approach is used to move the robot end-effector along a given reference trajectory solving the redundant inverse kinematics problem of the robot with respect to Cartesian motions.

The controller computes optimal joint velocities and joint positions for the next N_p prediction steps

$$\boldsymbol{V}^* = \{\boldsymbol{v}^*(0), \ldots, \boldsymbol{v}^*(N_\mathrm{p}-1)\} \tag{3}$$

$$\boldsymbol{Q}^* = \{\boldsymbol{q}^*(1), \ldots, \boldsymbol{q}^*(N_\mathrm{p})\} \tag{4}$$

based on the currently measured joint positions $\boldsymbol{q}(0) = \boldsymbol{q}_0$. The optimal joint velocities $\boldsymbol{v}^*(0)$ are applied to the robot. Then the optimization is repeated using the updated robot measurements.

For following the desired Cartesian trajectory $\boldsymbol{y}_\mathrm{t}(k)$, the objective function

$$\sum_{k=0}^{N_\mathrm{p}-1} F(\boldsymbol{v}(k), \boldsymbol{q}(k+1)) + E(\boldsymbol{q}(N_\mathrm{p})) \tag{5}$$

is minimized subject to the equality constraints

$$\boldsymbol{q}(k+1) = \boldsymbol{q}(k) + T_\mathrm{s}\boldsymbol{v}(k) \tag{6}$$

and the inequality constraints

$$\boldsymbol{q}_\mathrm{min} \leq \boldsymbol{q}(k+1) \leq \boldsymbol{q}_\mathrm{max} \tag{7}$$

$$\boldsymbol{v}_\mathrm{min}(k) \leq \boldsymbol{v}(k) \leq \boldsymbol{v}_\mathrm{max}(k). \tag{8}$$

for $k = 0, \ldots, N_\mathrm{p} - 1$. The equality constraints (6) contain the discretized robot model (1) with sampling time T_s. The inequality constraints (7) guarantee that the joint position limits are not exceeded. By choosing

$$\boldsymbol{v}_\mathrm{min}(k) = \max(-\boldsymbol{v}_\mathrm{max}, \boldsymbol{v}_0 - k \cdot T_\mathrm{s} \cdot \boldsymbol{a}_\mathrm{max}) \tag{9}$$

$$\boldsymbol{v}_\mathrm{max}(k) = \min(\boldsymbol{v}_\mathrm{max}, \boldsymbol{v}_0 + k \cdot T_\mathrm{s} \cdot \boldsymbol{a}_\mathrm{max}) \tag{10}$$

depending on the current joint velocity $\boldsymbol{v}_0$ and the maximum absolute value of the joint acceleration $\boldsymbol{a}_\mathrm{max}$, the inequality constraint (8) secures that the joint velocity limits and in the first step, that is applied to the robot, also the joint acceleration limits are met. $\boldsymbol{v}_\mathrm{max}$ is the maximum absolute value of the joint velocity due to the joint limits.

The objective function contains the cost function

$$F(\boldsymbol{v}, \boldsymbol{q}) = \boldsymbol{v}^\top \boldsymbol{Q}_v \boldsymbol{v} + \boldsymbol{q}^\top \boldsymbol{Q}_q \boldsymbol{q} + \boldsymbol{e}_\mathrm{t}^\top \boldsymbol{Q}_e \boldsymbol{e}_\mathrm{t} \tag{11}$$

and the end-penalty

$$E(\boldsymbol{q}) = \boldsymbol{q}^\top \boldsymbol{R}_q \boldsymbol{q} + \boldsymbol{e}_\mathrm{t}^\top \boldsymbol{R}_e \boldsymbol{e}_\mathrm{t} \tag{12}$$

with the trajectory following error

$$\boldsymbol{e}_\mathrm{t}(k) = \boldsymbol{f}(\boldsymbol{q}(k)) - \boldsymbol{y}_\mathrm{t}(k), \tag{13}$$

that gives the deviation of the resulting end-effector pose (according to the direct kinematics) from the reference pose. $\boldsymbol{Q}_v$, $\boldsymbol{Q}_q$, $\boldsymbol{Q}_e$, $\boldsymbol{R}_q$ and $\boldsymbol{R}_e$ are positive semi-definite diagonal matrices. The cost function $F(\boldsymbol{v}, \boldsymbol{q})$ penalizes high joint velocities to reduce the input energy and oscillations, deviations of the joint positions from the home position to keep the joints away from their constraints and to achieve more natural motions and the trajectory following error to complete the Cartesian task. The end penalty $E(\boldsymbol{q})$ is chosen similarly. It is only applied to the last prediction step and is added to improve stability.

The optimization is solved by nonlinear programming using a primal-dual interior point algorithm from the optimization library *IPopt* [14].

IV. Collision Avoidance

The Cartesian Nonlinear Model Predictive Controller presented in the previous Section is now extended to self-collision avoidance and prevention of collisions with static or dynamic obstacles.

A. Cartesian Control with Self-Collision Avoidance

During the execution of the Cartesian task, the resulting joint configurations must not lead to collisions of the robot with itself. Therefore, the NMPC approach of Section III-B is extended by further inequality constraints that prohibit self-collisions.

The robot geometry is approximated by n_R spheres. These spheres are described by their centers $\boldsymbol{p}_{\mathrm{R},i}(\boldsymbol{q}) \in \mathbb{R}^3$ and their radii $r_{\mathrm{R},i}$ $(i = 1, \ldots, n_\mathrm{R})$ and have to envelop all robot parts. Then, the distance between the centers of two robot spheres i and j that do not belong to the same link or to subsequent links has to meet the constraint

$$d^2_{\mathrm{R},i,j}(\boldsymbol{q}(k)) > d^2_{\mathrm{R},i,j,\mathrm{min}} \tag{14}$$

for $k = 1, \ldots, N_\mathrm{p}$ with

$$d^2_{\mathrm{R},i,j}(\boldsymbol{q}) = \left\| \boldsymbol{p}_{\mathrm{R},i}(\boldsymbol{q}) - \boldsymbol{p}_{\mathrm{R},j}(\boldsymbol{q}) \right\|^2 \tag{15}$$

and

$$d_{\mathrm{R},i,j,\mathrm{min}} = r_{\mathrm{R},i} + r_{\mathrm{R},j} \ . \tag{16}$$

Collisions between subsequent links are prevented by restricting the position range of the connecting joint or are generally impossible due to the robot geometry.

B. Cartesian Control with Obstacle Avoidance

Additionally, the robot redundancy is exploited to avoid collisions with obstacles based on the robot-obstacle distance.

The robot-obstacle collision avoidance uses the spherical approximation of the robot geometry introduced in Section IV-A as well. In order to achieve smooth distance computations for each optimization step, n_O spherical obstacles are considered with center $\boldsymbol{p}_{\mathrm{O},j} \in \mathbb{R}^3$ $(j = 1, \ldots, n_\mathrm{O})$ and radius r_O.

To avoid collisions, the squared distance between robot and obstacle sphere centers

$$d^2_{i,j}(\boldsymbol{q}) = \left\| \boldsymbol{p}_{\mathrm{R},i}(\boldsymbol{q}) - \boldsymbol{p}_{\mathrm{O},j} \right\|^2 \tag{17}$$

has to be greater than the squared minimum distance $d^2_{i,\mathrm{min}}$ with

$$d_{i,\mathrm{min}} = r_{\mathrm{R},i} + r_\mathrm{O} + d_\mathrm{s} \tag{18}$$

where d_s is a given safety distance that takes localization and sensing uncertainties into account. This is guaranteed by augmenting the inequality constraints (8), (7) and (14) with

$$d^2_{i,j}(\boldsymbol{q}(k)) > d^2_{i,\mathrm{min}} \tag{19}$$

for $k = 1, \ldots, N_\mathrm{p}$, $i = 1, \ldots, n_\mathrm{R}$ and $j = 1, \ldots, n_\mathrm{O}$.

These constraints keep the safety distance between the robot and the obstacles but allow the robot to move close to the obstacles. In the case of dynamic obstacles like humans, however, it is desirable to maximize the distance to obstacles as far as the Cartesian task execution is not impeded. Therefore, the cost function (5) is extended by a term that increases with decreasing distances between robot and obstacles,

$$\sum_{k=0}^{N_\mathrm{p}-1} F_\mathrm{O}(\boldsymbol{q}(k+1)) + E_\mathrm{O}(\boldsymbol{q}(N_\mathrm{p})) \tag{20}$$

with

$$F_\mathrm{O}(\boldsymbol{q}) = \sum_{i=1}^{n_\mathrm{R}} \sum_{j=1}^{n_\mathrm{O}} \frac{c_i}{d^2_{i,j}(\boldsymbol{q}) - d^2_{i,\mathrm{min}}} \tag{21}$$

$$E_\mathrm{O}(\boldsymbol{q}) = f_\mathrm{O} F_\mathrm{O}(\boldsymbol{q}) \ . \tag{22}$$

Due to this cost function extension, the robot is pushed away from obstacles while at the same time the robot end-effector follows the reference trajectory as close as possible.

C. Combined Obstacle Detection and Collision Avoidance

Using the 3D obstacle representation of Section II, each octree node that is occupied or occluded by an obstacle is considered as obstacle sphere where the sphere center corresponds to the node center and the sphere radius is $r_\mathrm{O} = \frac{\sqrt{3}}{2} r$ depending on the octree resolution r. But as considering all obstacle nodes in the control algorithm would lead to a too high computing effort, only the relevant obstacle spheres are selected.

The most relevant obstacles are those close to the robot. So two approaches are analyzed that use only obstacle points close to the robot. The first approach computes for each robot link the obstacle sphere with minimum distance to this link. Only this obstacle is considered for all robot spheres that belong to the corresponding link. In the second approach, the obstacle sphere with minimum distance is computed for each robot sphere separately. The controller uses then these pairs of robot and obstacle spheres for collision avoidance. Compared to the first approach, this approach is beneficial for large robot links like the mobile base that may be affected by several obstacle spheres. In contrast, the first approach is able to compute exacter distances as not necessarily the same robot description has to be used for distance computation and control. Instead, the distance computation can use, e.g., a CAD description or an approximation based on simple geometric shapes as cylinders and cuboids.

A further reduction of the number of constraints can be achieved by considering the minimum time that the robot needs to reach an obstacle. In [15] a method has been proposed to compute a reachability grid. The reachability grid contains in each grid cell an approximation of the minimum time T_min that the robot needs to reach this cell with at least one robot part based on the current joint configuration and the maximum velocities. Using this information, it is determined for each prediction step if a specific obstacle is reachable and therefore can be in collision with the robot. Only obstacle spheres with reachability time

$$T_{\mathrm{O},j,\mathrm{min}} \leq k \cdot T_\mathrm{s} + T_\mathrm{safety} \tag{23}$$

are considered in the inequality constraints (19) for prediction step k, where T_safety is a time interval, that considers the computing time of the reachability analysis and is added for safety reasons. Nevertheless, the non-reachable obstacle spheres are still used in the cost function (20), to achieve foresightful evasive motions.

V. Results

A. Experimental Setup

The proposed algorithms are applied to an omnidirectional mobile manipulator (KUKA OmniRob with LWR IV) with 10 degrees of freedom (DoF) that is shown in Fig. 2. The actuating variables of the robot are the joint and platform velocities. The joint positions of the arm are directly measured, the platform position and heading angle are derived from the platform odometry and a map-based localization framework. The platform is controlled by commanding the translational velocities v_x and v_y and the rotational velocity v_3 with respect to the current platform pose. The translational velocities v_x and v_y are computed by transforming the velocities v_1 and v_2 of the prismatic platform joints from the global frame to the current robot base frame according to the current platform rotation q_3,

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \cos(q_3) & \sin(q_3) \\ -\sin(q_3) & \cos(q_3) \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}. \tag{24}$$

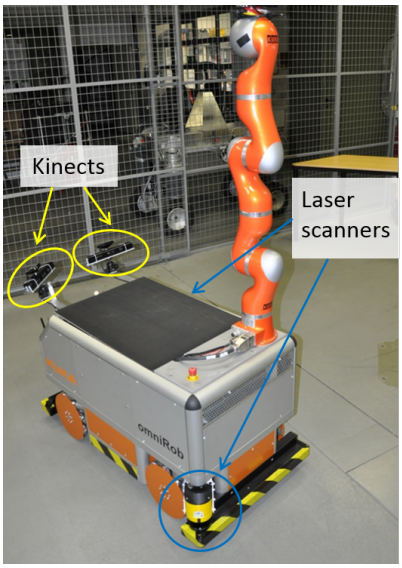


Fig. 2. Robot setup: KUKA OmniRob with LWR, equipped with two laser scanners and two Kinect sensors.

For the control algorithm a sampling time of 0.1 s and a prediction horizon of 1.0 s are used. The diagonal matrices of the cost function (11) and the terminal penalty (12) are chosen as follows:

$$\boldsymbol{Q}_v = \mathrm{diag}(\alpha_v, \alpha_v, \alpha_v, \alpha_v, \alpha_v, \alpha_v, \alpha_v, \alpha_v, \alpha_v, \alpha_v), \quad \alpha_v = 5$$
$$\boldsymbol{Q}_q = \mathrm{diag}(0, 0, 0, \alpha_q, \alpha_q, \alpha_q, \alpha_q, \alpha_q, \alpha_q, \alpha_q), \quad \alpha_q = 0.01$$
$$\boldsymbol{Q}_e = \mathrm{diag}(\alpha_\mathrm{p}, \alpha_\mathrm{p}, \alpha_\mathrm{p}, \alpha_\mathrm{o}, \alpha_\mathrm{o}, \alpha_\mathrm{o}, \alpha_\mathrm{o}), \quad \alpha_\mathrm{p} = 200, \quad \alpha_\mathrm{o} = 150$$
$$\boldsymbol{R}_q = \boldsymbol{0}$$
$$\boldsymbol{R}_e = 100\boldsymbol{Q}_e$$

Note that elements of $\boldsymbol{Q}_q$ belonging to the platform joints are zero, as it is not intended to keep the platform at its start position and orientation. In $\boldsymbol{Q}_e$ and $\boldsymbol{R}_e$, all position weights as well as all orientation weights are equal.

To avoid collisions, the cost function extension (21) is applied with $c_i = 0.1$ for all robot points i and f_O is chosen to be 20. The spherical approximation of the robot geometry for collision avoidance is shown in Fig. 3. The spheres describing the mobile base are significantly larger than the real robot geometry in order to guarantee a safety distance around the mobile base that is necessary due to the high mass and the possible high velocities of the mobile base.

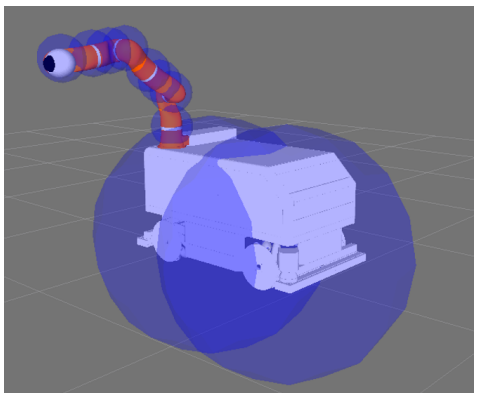


Fig. 3. Spherical approximation of the robot geometry for collision avoidance.

For monitoring the robot environment, the robot is equipped with two laser scanners and two Kinect sensors (see Fig. 2). The two laser scanners monitor a plane all around the platform near the ground floor. The two Kinect sensors observe the space around the manipulator.

The monitoring and control algorithms run on a 2.8 GHz Intel Core i7 processor. The robot filter [11] is GPU based.

The results are analyzed with regard to the Euclidean distance between the resulting end-effector position and the the reference position $\left\|\boldsymbol{y}_\mathrm{p} - \boldsymbol{y}_\mathrm{t,p}\right\|$ and the angle between the resulting end-effector orientation and the reference orientation $\arccos(2\left\langle \boldsymbol{y}_\mathrm{o}, \boldsymbol{y}_\mathrm{t,o} \right\rangle^2 - 1)$, where the resulting end-effector pose is computed based on the direct kinematics (2) and the currently measured joint positions. $\langle \cdot, \cdot \rangle$ denotes the scalar product of two quaternions. Furthermore, the minimum distance between the robot surface and the obstacle representation and the computing time needed per control step are analyzed.

B. Simulation Results

In order to compare the different possibilities how obstacles are treated for collision avoidance, first a simulation with static obstacles is shown. The simulated robot corresponds to the omnidirectional mobile manipulator described in Section V-A. The simulation scenario is shown in Fig. 4. During the simulation, the robot end-effector follows the green reference trajectory, that lies between red obstacle spheres. The computed minimum distance between the robot links and the obstacles is visualized as yellow lines.

The resulting deviation of the end-effector position and orientation from the reference trajectory, the resulting minimum distance between the robot and the obstacles, as well as the required computation time per control step are analyzed for several test cases (Fig. 5):

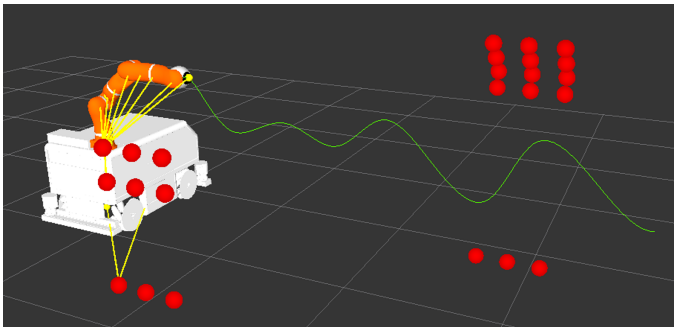


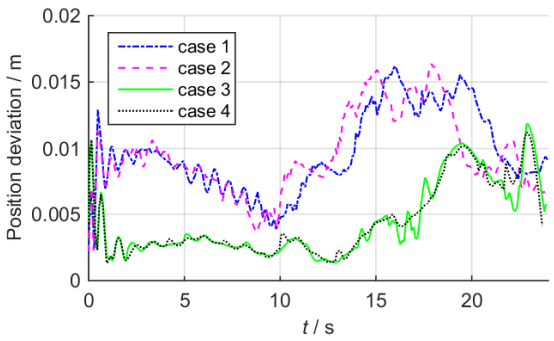
Fig. 4. Simulation scenario of a mobile manipulator executing a trajectory following task with obstacle avoidance. Trajectory: green line, obstacles: red spheres, distances between robot and obstacles: yellow lines.

- Case 1: All obstacle spheres with minimum distance to one of the robot links are considered for collision avoidance with all robot spheres.
- Case 2: The same obstacle spheres are considered but the number of optimization constraints due to obstacles is reduced by taking into account, if an obstacle sphere is reachable by the robot in a specific prediction step.
- Case 3: The number of considered obstacles per robot sphere is reduced, by using for each robot point only the obstacle sphere with closest distance to the corresponding link. This affects both the number of constraints and the cost function.
- Case 4: For each robot sphere, the obstacle with minimum distance to the robot sphere is considered for collision avoidance.

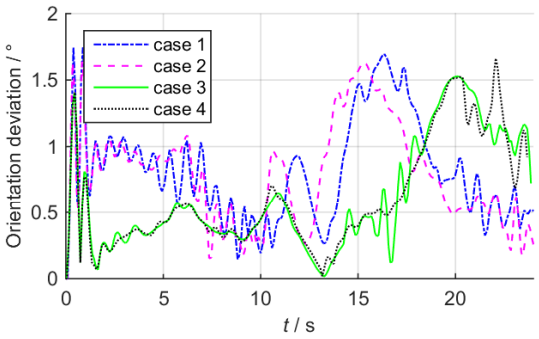
It can be seen, that in cases 3 and 4, the minimum robot obstacle distance decreases compared to the first two cases (Fig. 5(c)), as less obstacle points are used in the cost function. Accordingly, the maximum position deviation is in the first two cases higher (1.6 cm) than in the cases 3 and 4 where it amounts to 1.1 cm (Fig. 5(a)). But the position deviation is in both cases acceptable. The decisive factor is the computing time per control step. With the reduced number of considered obstacle spheres per robot sphere in cases 3 and 4, the computing time decreases significantly and lies in a suitable range for a sampling time of 0.1 s (Fig. 5(d)).

Taking the obstacle reachability into account in order to reduce the number of optimization constraints has only a small effect in case 2. Further experiments have shown, that combining the reachability analysis with the reduced number of obstacle spheres per robot point in test cases 3 and 4 does not further reduce the computing time.

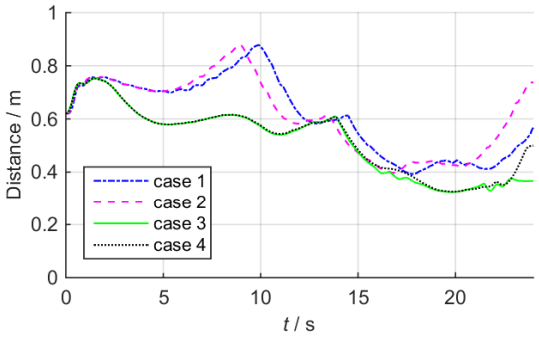
Comparing cases 3 and 4 shows, that the approach in case 4, that considers for each robot sphere the obstacle sphere with minimum distance, achieves slightly better results. The deviation from the reference pose contains less oscillations (Fig. 5(a) and 5(b)) and the distance to obstacles is equal to or even larger than the distance in case 3 (Fig. 5(c)). The reason is, that in the narrow passage between the left and right group of obstacles the mobile base alternates in moving closer



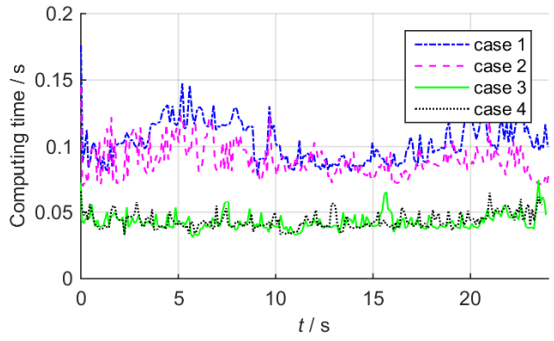
(a) End-effector position deviation



(b) End-effector orientation deviation



(c) Minimum robot-obstacle distance



(d) Computing time for one control step

Fig. 5. Simulation results of a mobile manipulator executing a trajectory following task with obstacle avoidance.

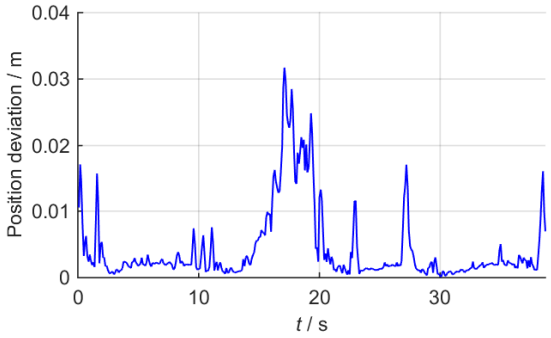
to the left and the right obstacle group if only one distance is computed for the complete base. Therefore, the following experiment is conducted with the configuration of case 4.

C. Experimental Results

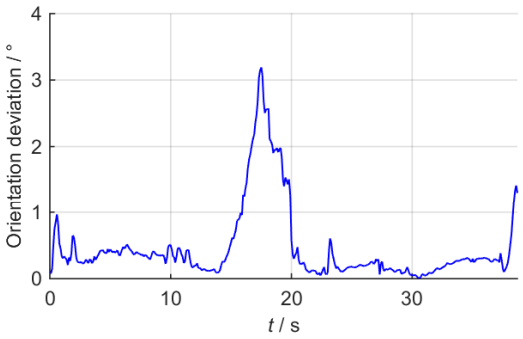
The Cartesian Nonlinear Model Predictive Control approach with collision avoidance is now applied to the real mobile manipulator described in Section V-A. Fig. 7 shows snapshots of the experimental scenario including the obstacle representation and the resulting robot motion. The mobile manipulator has to follow the reference trajectory of the end-effector pose shown as green line. During the task execution, a human is walking near the robot and grasping into the robot path.

Using the sensor data from the two laser scanners and the two Kinect sensors, a 3D octree representation of the obstacles in the close robot environment and their occlusions are computed with a resolution of 10 cm with the method presented in Section II. The octree nodes that contain obstacle measurements are visualized red, the occluded nodes rose. The octree nodes representing the human arm can be seen on the left side of the robot in Fig. 7b-f. On the right side of the robot, the representation of a wall detected by the laser scanners is visible. The minimum distances between the robot spheres and the obstacle octree are shown as yellow lines. For each robot sphere, the octree node with minimum distance is considered.

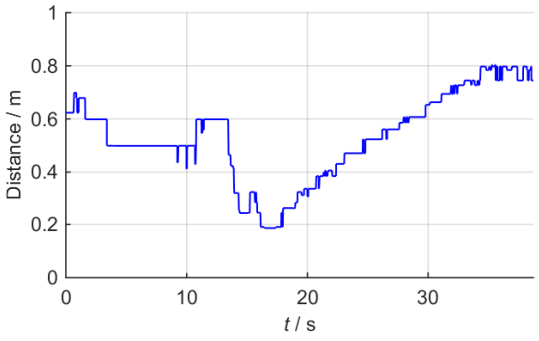
Despite the moving obstacle that is actively approaching the robot, the robot is able to follow the reference trajectory without any collision due to the evasive motions performed by the controller. The evasive motion can be seen for example in the platform motion (Fig. 8): when the human approaches, the platform moves backwards in order to allow the robot manipulator to increase the distance to the human. As shown in Fig. 6, the maximum deviation of the achieved end-effector position from the reference position during the experiment is 3.2 cm, the maximum orientation deviation is 3.1 °. The maximum deviation occurs, when the human moves towards the robot and the robot has to evade the human. Due to the evasive motion, the minimum distance between the robot and the obstacles never becomes smaller than 27 cm. The computing time for one control step remains lower than 60 ms during the complete experiment, which is absolutely sufficient for the sampling time of 100 ms.



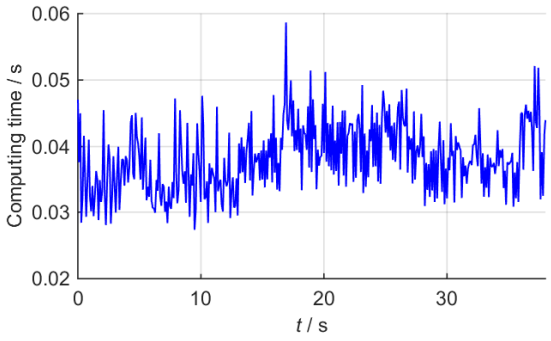
(a) End-effector position deviation



(b) End-effector orientation deviation



(c) Minimum robot-obstacle distance



(d) Computing time for one control step

Fig. 6. Experimental results of a mobile manipulator executing a trajectory following task with dynamic obstacle avoidance.

VI. Conclusions

In this work, a Cartesian Nonlinear Model Predictive Controller for controlling the end-effector pose of fixed-base and mobile manipulators has been extended by self-collision avoidance and collision avoidance with static and dynamic obstacles. The obstacles in the close robot environment are detected by 3D workspace monitoring based on depth sensors mounted on the robot. Realtime performance of the collision avoidance extension has been achieved by considering in each control step only the most relevant obstacles. Different collision avoidance configurations have been compared by means of simulation. The Cartesian controller with the

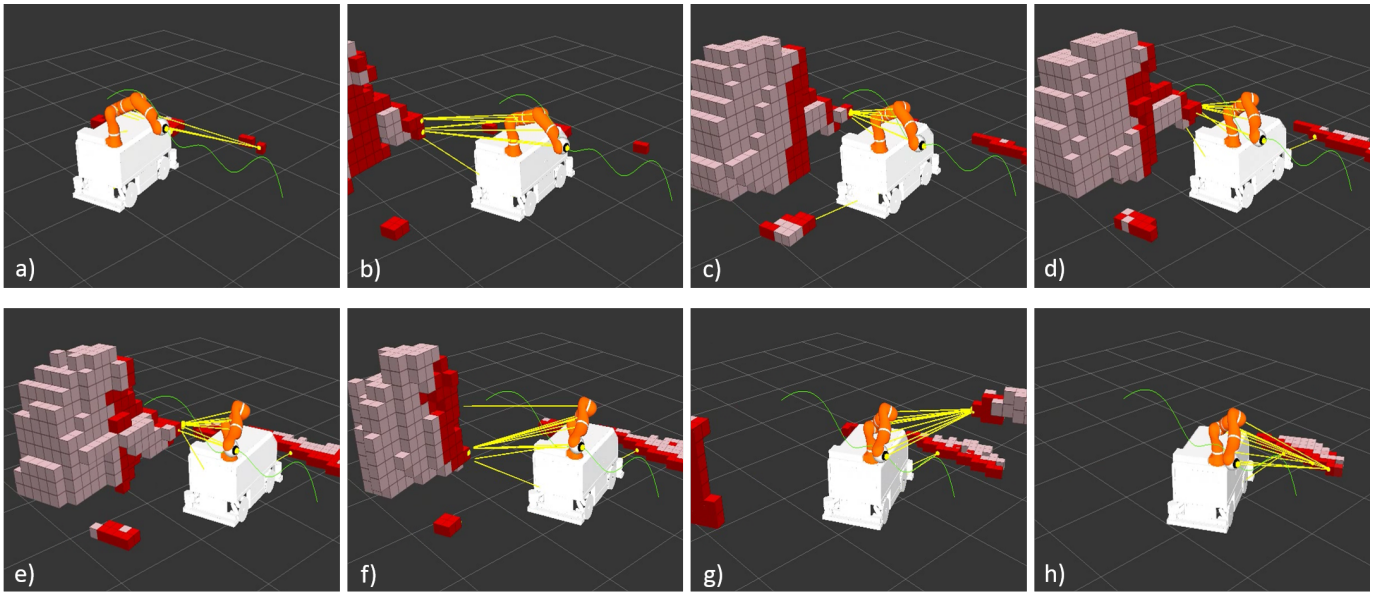


Fig. 7. Snap-shot sequence of a mobile manipulator following a Cartesian reference trajectory (green line) with dynamic obstacle avoidance. Obstacles are represented in an octree structure based on depth sensor measurements with occupied nodes as red cubes and occluded nodes as rose cubes. Yellow lines give the distance between the robot points and the obstacles.

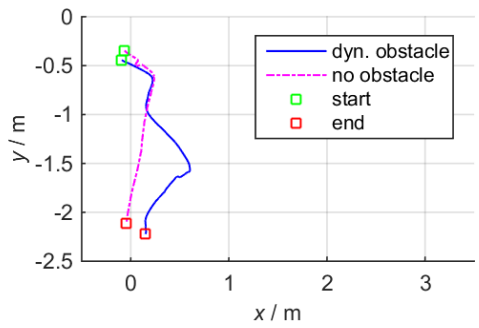


Fig. 8. Comparison of the platform position during the trajectory following experiment with dynamic obstacle avoidance as shown in Fig. 6 and Fig. 7 and without dynamic obstacle.

most promising collision avoidance configuration has been successfully applied to a real mobile manipulator with 10 DoF avoiding a moving human.

Acknowledgment

This work has been funded by the European Commission's 7th Framework Programme as part of the project SAPHARI under grant agreement ICT-287513.

The author thanks her colleague Christian Frese for the inspiring discussions and comments on the presented work.

References

[1] Lihui Wang, *Collaborations towards adaptive manufacturing*, in IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 14–21, 2012.

[2] O. Khatib, *Real-time obstacle avoidance for manipulators and mobile robots*, in IEEE International Conference on Robotics and Automation, pp. 500–505, 1985.

[3] H. G. Tanner, S. G. Loizou, and K. J. Kyriakopoulos, *Nonholonomic navigation and control of cooperating mobile manipulators*, in IEEE International Conference on Robotics and Automation, pp. 53–64, 2003.

[4] Pei-Wen Wu, Yu-Chi Lin, Chia-Ming Wang and Li-Chen Fu, *Grasping the object with collision avoidance of wheeled mobile manipulator in dynamic environments*, in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5300–5305, 2013.

[5] Y. Kitamura, F. Kishino, T. Tanaka, and M. Yachida, *Real-time path planning in a dynamic 3-D environment*, in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 925–931, 1996.

[6] M. P. Polverini, A. M. Zanchettin, and P. Rocco, *Real-time collision avoidance in human-robot interaction based on kinetostatic safety field*, in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4136–4141, 2014.

[7] F. Flacco, T. Kroger, A. De Luca, and O. Khatib, *A depth space approach to human-robot collision avoidance*, in IEEE International Conference on Robotics and Automation (ICRA), pp. 338–345, 2012.

[8] Huatao Zhang, Yunyi Jia, Ning Xi, and Aiguo Song, *Obstacle avoidance for mobile manipulation by real-time sensor-based redundancy resolution*, in IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 2369–2374, 2012.

[9] A. Zube, *Cartesian Nonlinear Model Predictive Control of Redundant Manipulators Considering Obstacles*, in IEEE International Conference on Industrial Technology (ICIT), 2015.

[10] A. Fetzner, C. Frese, and C. Frey, *A 3D Representation of Obstacles in the Robots Reachable Area Considering Occlusions*, in International Symposium on Robotics (ISR/Robotik), pp. 1–8, 2014.

[11] N. Blodow: *Realtime URDF Filter*. [Online]. Available: https://github.com/blodow/realtime_urdf_filter.

[12] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss and W. Burgard: *OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees*, Autonomous Robots, 2013.

[13] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, Springer, 2008.

[14] A. Wächter and L. T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, Mathematical Programming, pp. 25–57, 2006.

[15] P. Anderson-Sprecher, and R. Simmons, *Voxel-based motion bounding and workspace estimation for robotic manipulators*, in IEEE International Conference on Robotics and Automation (ICRA), pp. 2141–2146, 2012.