Abstract—For redundant fixed 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. In this contribution, a Nonlinear Model Predictive Control (NMPC) approach is proposed to move the end-effector of the robot to a desired pose (3D position and orientation), along a geometric reference path or along a reference trajectory while considering obstacles. Due to the underlying general robot model, the control algorithm is applicable to both fixed and mobile manipulators, which is shown by means of simulations of a 7 DoF fixed manipulator and of a 10 DoF mobile manipulator.

I. INTRODUCTION

With regard to shared human-robot workspaces and cluttered environments, the use of redundant robots is desirable. As they have more degrees of freedom (DoF) than needed to execute their given task, the redundant degrees of freedom can be exploited to achieve an additionally desired robot behavior, e.g., avoiding collisions with humans or other obstacles, while the robot continues performing its original task.

The robot task is often specified in the Cartesian space. The end-effector has to move to a desired Cartesian pose (3D position and orientation), has to follow a geometric reference path, e.g., defined by a sequence of poses, or has to follow a reference trajectory that defines the desired Cartesian pose as a function of time. That means, the robot has to be controlled in order to fulfill the Cartesian task by applying suitable joint configurations or velocities to a secondary position or velocity controller that is usually provided by the robot manufacturer. A common approach is to transform the Cartesian reference poses into joint configurations by solving the inverse kinematics problem and then performing control strategies in the joint space. But in the case of redundant robots, this procedure has the disadvantage, that there is no unique solution for the inverse kinematics problem. Solutions resulting from (weighted) pseudo-inverses of the direct kinematics as presented in [1] are often not optimal with regard to the task execution and are not suitable for collision avoidance.

To overcome this problem, this contribution proposes a Nonlinear Model Predictive Control (NMPC) algorithm that accepts Cartesian reference paths or trajectories of the desired 3D end-effector position and orientation as input and computes optimal joint positions and velocities. The controller is based on a generic robot model, so that the controller is applicable to both fixed manipulators and mobile manipulators. Joint position and velocity constraints are directly considered in the control algorithm. Additionally to the Cartesian task execution, collision avoidance is achieved by using a cost function and constraints that depend on the distance between the robot and the obstacles. The presented control algorithm is also easily modifiable to realize other desired behaviors by adding additional terms to the objective function.

This paper is organized as follows: After an overview of related work in Section II, the underlying robot model is described in Section III. In Section IV, the Cartesian motion task is formally defined as trajectory and path following problem. The NMPC algorithm that solves the trajectory or path following problems is discussed in Section V with an addition for obstacle avoidance. Simulation results for a 7 DoF manipulator and a 10 DoF mobile manipulator are presented in Section VI. The paper closes with the conclusion in Section VII.

II. RELATED WORK

Model predictive control methods are often applied to solve the trajectory following problem of mobile platforms [2]–[4]. Usually, the 2D platform position and the heading angle are controlled in order to follow a given reference trajectory based on a kinematic model of the robot.

With respect to robot manipulators, model predictive control is often reported to be used for joint position or velocity control [5]–[10]. In these examples optimal joint torques are computed in order to follow a desired trajectory in the joint space based on a dynamic robot model. In some cases, the joint trajectory is reported to be obtained from a Cartesian reference trajectory by solving the inverse kinematics. In [11], a goal joint configuration is given instead of a joint trajectory. The robot manipulator is controlled in order to reach this joint configuration while avoiding collisions with other robots. The robot geometries are modeled by line-swept sphere volumes. Collision avoidance is realized by adding collision constraints to the optimization problem.

In [12], model predictive control for robot manipulators is extended in order to solve the path following problem for a given joint position path without time dependency. Therefore, the velocity along the path is optimized together with the robot torques. The method is applied to a two-link planar arm configuration.

For mobile manipulators, only a few model predictive control approaches exist. In [13] model predictive control is proposed to control the end-effector position of a mobile manipulator to reach the desired reference position. The considered mobile manipulator consists of an omni-directional platform and two planar links. In [14] the end-effector position
of a two-link planar mobile manipulator is controlled in consideration of obstacles. Collision avoidance is achieved by optimization constraints that incorporate the distance between the spherical obstacles and the platform or link centers.

By contrast, this contribution uses an NMPC approach to control the 3D end-effector pose of both fixed and mobile manipulators based on a general robot model. For collision avoidance, the robot geometry is approximated by spherical bounding volumes and obstacles are treated as 3D points to achieve a smooth proximity computation. The distances between the robot and obstacles do not only affect the optimization constraints but are also maximized by means of additional objective function terms.

III. ROBOT MODEL

Generally, a (mobile) manipulator that can be modeled as a chain of rigid links connected by \( n \) revolute or prismatic joints is described by the following kinematic model disregarding accelerations

\[
\begin{align*}
\dot{q} &= v \\
y &= f(q)
\end{align*}
\]

with the revolute or prismatic joint positions \( q \in \mathbb{R}^n \), the joint velocities \( v \in \mathbb{R}^n \) and the end-effector pose \( y = [y_p, y_o]^T \in \mathbb{R}^m \). The end-effector pose is composed of the position \( y_p \) and the orientation \( y_o \). The nonlinear function \( f(q) \) maps the joint configuration \( q \) into the Cartesian space \( \mathbb{R}^m \) and is called the direct kinematics of the robot.

A common approach is to describe the end-effector orientation \( y_o \) by three Euler angles, which yields together with the three-dimensional position \( y_p \in \mathbb{R}^3 \) to \( m = 6 \). Euler angles have the advantage that they are a minimal representation of the orientation, but they suffer from coordinate singularities. Therefore, in this contribution the Cartesian orientation is represented by unit quaternions, which are a singularity free representation of orientations [15], and the Cartesian space has the dimension \( m = 7 \).

The direct kinematics \( f(q) \) is derived from the Denavit-Hartenberg convention. Using the Denavit-Hartenberg parameters \( \alpha, \theta, a \) and \( d \) according to [16], the transform between the link frame \( i \) and the link frame \( i-1 \) that are connected by joint \( i \) is given by

\[
T_{i-1}^i = \begin{bmatrix}
    c_{\theta_i} & -s_{\theta_i} & c_{\alpha_i} & s_{\alpha_i} & a_i c_{\alpha_i} \\
    s_{\theta_i} & c_{\theta_i} & -s_{\alpha_i} & c_{\alpha_i} & a_i s_{\alpha_i} \\
    0 & s_{\alpha_i} & c_{\alpha_i} & 0 & d_i \\
    0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

with the abbreviations \( c_\theta = \cos(\theta), s_\theta = \sin(\theta), c_\alpha = \cos(\alpha) \) and \( s_\alpha = \sin(\alpha) \). If the joint \( i \) is revolute, \( \theta_i \) depends on the joint position with \( \theta_i(q_i) = \theta_{i,0} + q_i \). If the joint is prismatic, \( d_i(q_i) = d_{i,0} + q_i \) holds. All other parameters are constant. The homogeneous transformation between the end-effector and the base frame is computed as product of the homogeneous transformations \( T_{i-1}^i(q_i) \) associated to joint \( i \) \((i = 1, \ldots, n)\)

\[
T_0^n(q) = T_0^1(q_1) \cdot T_1^2(q_2) \cdots \cdot T_{n-1}^n(q_n).
\]

The end-effector position is obtained from the last column of \( T_0^n(q) \).

\[
\begin{bmatrix}
y_p(q) \\
y_o(q)
\end{bmatrix} = T_0^n(q) \cdot [0 \\ 0 \\ 0 \\ 1]^T.
\]

The end-effector orientation \( y_o(q) \) is computed according to the conversion of rotation matrices to quaternions given in [15].

For robot manipulators, determining the Denavit-Hartenberg parameters is well studied and can be found e.g., in [16]. But the procedure is simply extensible to mobile manipulators by adding two prismatic joints for the translational platform motion and one revolute joint for the platform rotation. This is sufficient for holonomic (omni-directional) platforms. In the case of a non-holonomic platform, the non-holonomic properties are described by additional constraints, which can be handled by the presented Nonlinear Model Predictive Controller as well. Hence, a generic modeling framework is available that can be applied to both fixed manipulators and mobile manipulators.

IV. PROBLEM FORMULATION

In this contribution, robot tasks are considered that are defined in the Cartesian space by a desired reference pose of the end-effector, a reference path, e.g., specified by a sequence of poses, or a reference trajectory. These tasks are examined here as Cartesian path following or trajectory following problems.

If the robot task consists in following the given Cartesian trajectory \( y(t) \in \mathbb{R}^m \), the goal of the trajectory controller is to minimize the deviation of the end-effector pose from the reference trajectory, which is described by the trajectory following error

\[
e_t(t) = y(t) - y_k(t).
\]

The problem of moving the end-effector to a single reference pose \( y_t \) is treated as a special case of the trajectory following problem with \( y_k(t) = y_t = \text{const.} \).

Often no trajectory with time dependency is available and the robot task consists in following a geometric reference path, e.g., given by discrete path points. Then, the necessary time dependency is modeled by introducing a path parameter \( \theta(t) \) as proposed in [12]. The path parameter describes the current pose on the path \( p(\theta(t)) \in \mathbb{R}^m \) with \( \theta = \theta_{\text{min}} \) at the start point and \( \theta = \theta_{\text{max}} \) at the end point of the path. The path following goal is to minimize the deviation of the end-effector pose from the reference path, which is described by the path following error

\[
e_p(t) = y(t) - p(\theta(t)).
\]

The evolution of \( \theta(t) \) is a-priori unknown. It is assumed, that the reference path is twice continuously differentiable. If the path is given by a sequence of target poses \( p_i \) \((i = 1, \ldots, n_{\text{path}})\), a smooth path can be obtained by interpolating the path points using cubic splines.

V. NONLINEAR MODEL PREDICTIVE CONTROL

To solve the above-described trajectory and path following problems, a Nonlinear Model Predictive Control (NMPC) approach is proposed based on the discretized robot model

\[
q(k + 1) = q(k) + T_k \cdot v(k) \]

\[
y(k) = f(q(k))
\]
with sample time $T_s$. NMPC optimizes the system input and state sequences

$$V = \{v(0), \ldots, v(N_p - 1)\}$$

$$Q = \{q(1), \ldots, q(N_p)\}$$

for the next $N_p$ time steps based on the currently measured joint positions $q(0) = q_0$ by minimizing a cost function containing the trajectory following error or the path following error respectively. The optimized input $v^*(0)$ is applied to the system. Then, the optimization is repeated using the next measurement.

The formulation of the optimization problem for trajectory and path following control is presented in the following sections. In both cases, the optimization problem is solved by nonlinear programming (NLP). Here, a primal-dual interior-point algorithm with a filter line-search method from the optimization library IPOPT [17] is used, that exploits the sparse structure of NMPC optimization problems.

### A. Optimization Problem for Trajectory Control

For following the desired Cartesian trajectory $y_d$, the optimal joint velocity and joint position sequences are computed by minimizing the cost function

$$\sum_{k=0}^{N_p-1} F(v(k), q(k+1)) + E(q(N_p)) \rightarrow \min_{V,Q}$$

subject to the equality constraints

$$q(k + 1) - q(k) - T_s v = 0$$

and the inequality constraints

$$-v_{\text{max}} \leq v(k) \leq v_{\text{max}}$$

$$q_{\text{min}} \leq q(k + 1) \leq q_{\text{max}},$$

for $k = 0, \ldots, N_p - 1$. The equality constraints (13) contain the system model (8). The inequality constraints (14)–(15) guarantee that the joint velocity and position limits are met.

The terms of the cost function are

$$F(v,q) = v^T Q_v v + q^T Q_q q + e_t^T Q_e e_t$$

$$E(q) = q^T R_q q + e_t^T R_e e_t$$

with the trajectory following error $e_t$ according to (6) and (9). $Q_v, Q_q, R_q$ and $R_e$ are positive semi-definite diagonal matrices. With $F(v,q)$, high joint velocities, deviations of the joint positions from the home position and the trajectory following error are penalized. Penalizing the trajectory following error is needed to fulfill the trajectory following task. Penalizing the joint velocities avoids unnecessary high input energy and oscillations. Avoiding high joint position values keeps the joints away from the joint constraints and results in more natural motion. The term $E(q)$ is only applied to the last prediction step and is added to improve stability.

### B. Optimization Problem for Path Control

Given a path instead of a trajectory, the reference pose at a specific time is unknown, as the evolution of the path parameter $\theta(t)$ is not given a-priori. Therefore, the path parameter is optimized together with the joint positions and velocities of the robot based on a system model that describes the path parameter dynamics. In this contribution the linear model

$$\theta(k + 1) = \theta(k) + T_s v_\theta$$

with the path parameter velocity $v_\theta$ is used. Hence, the optimization problem for trajectory control is augmented to find an optimal solution for the sequences

$$V = \{v(0), \ldots, v(N_p - 1)\}$$

$$Q = \{q(1), \ldots, q(N_p)\}$$

$$V_\theta = \{v_\theta(0), \ldots, v_\theta(N_p - 1)\}$$

$$\Theta = \{\theta(1), \ldots, \theta(N_p)\}$$

that minimizes the cost function

$$\sum_{k=0}^{N_p-1} F(v(k), q(k+1), v_\theta(k), \theta(k+1)) + E(q(N_p), \theta(N_p))$$

subject to

$$q(k + 1) - q(k) - T_s v = 0$$

$$\theta(k + 1) - \theta(k) - T_s v_\theta = 0$$

and

$$-v_{\text{max}} \leq v(k) \leq v_{\text{max}}$$

$$q_{\text{min}} \leq q(k + 1) \leq q_{\text{max}}$$

$$v_\theta_{\text{min}} \leq v_\theta(k) \leq v_\theta_{\text{max}}$$

$$\theta_{\text{min}} \leq \theta(k + 1) \leq \theta_{\text{max}}$$

for $k = 0, \ldots, N_p - 1$. The terms of the cost function are

$$F(v,q,v_\theta,\theta) = v^T Q_v v + q^T Q_q q + e_t^T Q_e e_t$$

$$+ q_\theta^T v_\theta^2 - q_\theta^T q_\theta$$

$$+ e_\theta^T e_\theta$$

$$+ v_\theta^T v_\theta - \theta_{\text{max}}^2$$

with $q_{\theta,\text{min}} \geq 0$, $q_\theta \geq 0$, $r_\theta \geq 0$ and the positive semi-definite diagonal matrices $Q_v, Q_q, Q_e, R_q$ and $R_e$. The path following error $e_p$ is computed according to (7) and (9). Penalizing the distance of the path parameter from $\theta_{\text{max}}$ is necessary to enforce approaching the end of the path. With $v_{\theta,\text{min}} \geq 0$ only forward motions on the path are possible.

### C. Considering Obstacles

During task execution, the robot redundancy can be exploited to avoid collisions with obstacles. Therefore, the above presented cost functions are extended in order to additionally increase the distance between the robot and the obstacles.

For smooth distance computations, distances between test points on the robot $p_{R,i}(q) \in \mathbb{R}^3$ ($i = 1, \ldots, n_R$) and obstacle points $p_{O,j} \in \mathbb{R}^3$ ($j = 1, \ldots, n_O$) are considered,

$$d_{i,j}^2(q) = \|p_{R,i}(q) - p_{O,j}\|^2.$$
The position of each robot test point with respect to the base frame depends on the joint configuration $q$ and is computed using the kinematic robot model according to

$$\begin{bmatrix} p_{R,i}(q) \\ 1 \end{bmatrix} = T_0^l(q) \begin{bmatrix} p_{l,i} \\ 1 \end{bmatrix}$$ (33)

where the point $i$ belongs to link $l$ and $p_{l,i}$ is its position in the corresponding link frame.

The obstacle points are updated in each control step, when new measurements are available, in order to take obstacle motions into account. But during optimizing the input and state variables for one control step, the obstacles are assumed to be static over the full prediction horizon, as no prediction of the obstacle motion is available. If the obstacle motion is predicted using advanced monitoring strategies, the prediction can also be incorporated in the control algorithm.

The cost function (12) or (23) is extended by a term that penalizes the squared distances between the robot and the obstacle points.

$$\sum_{k=0}^{N_p-1} F_O(q(k + 1)) + E_O(q(N_p))$$ (34)

For $F_O$ and $E_O$ different functions are possible. They have to be twice continuously differentiable and have to increase for small squared distances $d_{i,j}^2(q)$. In this work the terms are chosen to be

$$F_O(q) = \sum_{i=1}^{n_R} \sum_{j=1}^{n_O} c_i \frac{d_{i,j}^2(q) - d_{i,\text{min}}^2}{d_{i,\text{min}}^2}$$ (35)

$$E_O(q) = r_0 F_O(q)$$ (36)

with the following characteristic: For $c_i > 0$, the function $c_i/(d_{i,j}^2 - d_{i,\text{min}}^2)$ goes to infinity for $d_{i,j}^2 \to d_{i,\text{min}}^2$ and it goes to zero when $d_{i,j}^2$ increases. $d_{i,\text{min}}$ is the minimum allowed distance between the robot point $i$ and an obstacle point. If $d_{i,j}^2 < d_{i,\text{min}}^2$, the term $F_O$ is set to infinity.

Considering the quadratic distances between the robot and obstacles in the objective function results in pushing the robot away from obstacles while at the same time the robot end-effector is controlled to follow the reference pose as close as possible. But there is no guarantee, that the optimized robot configurations are not in collision with an obstacle. This guarantee can be achieved by introducing the following constraints:

$$d_{i,j}^2(q(k)) - d_{i,\text{min}}^2 \geq 0$$ (37)

for $k = 1, \ldots, N_p$, $i = 1, \ldots, n_R$ and $j = 1, \ldots, n_O$. That means that the distance between the robot and the obstacle points has always to be greater or equal to a specified safety distance $d_{i,\text{min}}$. The drawback of this approach is, that the number of nonlinear constraints increases significantly and therefore solving the optimization problem becomes more expensive.

Using only the obstacle constraints (37) without the cost function terms (34) is also possible. But then, the robot may move close to the obstacles keeping only the requested minimum distance, even if motions with higher distances are possible. But especially in the case of dynamic obstacles, it is desirable to increase the distance between the robot and the obstacles as far as the task execution is not impeded.

The drawback of this approach is, that the number of nonlinear constraints increases significantly and therefore solving the optimization problem becomes more expensive.

Using only the obstacle constraints (37) without the cost function terms (34) is also possible. But then, the robot may move close to the obstacles keeping only the requested minimum distance, even if motions with higher distances are possible. But especially in the case of dynamic obstacles, it is desirable to increase the distance between the robot and the obstacles as far as the task execution is not impeded.

For the consideration of obstacles, eight test points on the robot are chosen with a minimum distance to obstacle points of 0.2 m. These robot points are visualized in Fig. 1(a) as blue spheres with radii according to the minimum distances.

The robot manipulator is controlled to move from a start configuration to a reference pose by using the joint velocities as actuating variables. A sample time of $T_s = 0.1$ s and $N_p = 10$ prediction steps are used, which leads to a prediction horizon of 1 s. In Fig. 1(a) the start configuration of the example motion is shown. Without considering obstacles the manipulator successfully reaches the reference pose with the joint configuration shown in Fig. 1(b).

If the two obstacle points that are visualized in Fig. 1 as green spheres are considered in the control algorithm, the manipulator also reaches the reference pose but with a joint configuration with higher distances between the robot and the obstacle points (see Fig. 1(c)).

In both cases the maximum computing time for one control step is lower than 70 ms using an 2.8 GHz Intel Core i7 processor without parallelization of the control algorithm.

B. Mobile Manipulator with Reference Trajectory

In the second example, the developed method is applied to a simulated omni-directional mobile platform equipped with a 7 DoF manipulator (KUKA OmniRob with LWR IV). The
whole mobile manipulator has 10 DoF and is highly redundant with regard to Cartesian reference poses. The robot is modeled according to the Denavit-Hartenberg convention as a chain of revolute and prismatic joints. Two prismatic joints $q_1$ and $q_2$ describe the platform translation in $x$- and $y$-direction with respect to a global frame. A revolute joint $q_3$ models the platform rotation about the vertical robot $z$-axis.

On the real robot, the values for the platform joints $q_1, \ldots, q_3$ cannot be directly measured but are obtained from the platform localization based on a reference map and the platform odometry. The platform is controlled by commanding the translational velocities $v_x$ and $v_y$ and the rotational velocity $v_z$ 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 based on the current platform rotation $q_3$.

\[
\begin{bmatrix}
  v_x \\
  v_y 
\end{bmatrix} = \begin{bmatrix}
  \cos(q_3) & \sin(q_3) \\
  -\sin(q_3) & \cos(q_3)
\end{bmatrix} \begin{bmatrix}
  v_1 \\
  v_2
\end{bmatrix}.
\] (38)

For collision avoidance, ten test points are chosen on the robot. The minimum distance for the two points describing the platform is set to 0.6 m and the points on the robot arm have a minimum distance of 0.2 m. In Fig. 2, the corresponding volume approximation is visualized by blue spheres.

The mobile manipulator is controlled to follow the end-effector pose trajectory shown as yellow line in Fig. 2. Again, a sample time of $T_s = 0.1$ s and $N_p = 10$ prediction steps are used. The control parameters are chosen in order to achieve a good collision avoidance behavior. The trajectory following accuracy is of less importance. The diagonal elements of the matrices in the cost function are $Q_{v,ii} = 1.0$, $Q_{q,ii} = 0.01$ for manipulator joints, $Q_{q,ii} = 0$ for platform joints, $Q_{e,ii} = 100$, $R_{q,ii} = 0$ and $R_{e,ii} = 1000$. The obstacle cost function parameters are $c_1 = 1.0$ for platform points and $c_1 = 0.1$ for manipulator points.

The resulting end-effector position in the $y$-$z$-plane can be seen in Fig. 3(a). In the first case (blue line), the trajectory following control is performed without obstacles. The end-effector follows the reference trajectory well with a maximum deviation of 3.9 cm. The computing time for one control step is below 20 ms except for the first optimization, when no appropriate start values are available (see Fig. 3(b)).

The six obstacles that are visualized as green spheres in Fig. 2 are considered by the objective function in case 2 (green line) and by both the objective function and the constraints in case 3 (magenta line). In both cases the deviation of the resulting end-effector pose from the reference trajectory increases compared to case 1 (see Fig. 3(a)), especially near the obstacles for $y \in [2.5 \text{ m}, 3.2 \text{ m}]$. But the mobile manipulator successfully moves through the obstacles without any collision and reaches the end of the trajectory with acceptable deviations. The computing time in case 2 only slightly increases compared to case 1. In case 3, the computing time is significantly higher due to the high number of constraints. But even close to obstacles ($t > 20 \text{ s}$) the computing time is acceptable with values below 80 ms.

In case 2 and 3, the obstacles shown in Fig. 4 are considered in the objective function or in both the objective function and the constraints (case 3).

C. Mobile Manipulator with Reference Path

In this example, the mobile manipulator described in Section VI-B is controlled to follow a reference path. The reference path corresponds to the reference trajectory in the previous section without time information and is shown in Fig. 4.

The results are shown in Fig. 5. In case 1, no obstacles are considered. The end-effector follows the reference path with only small deviations below 1.0 cm. As no difficulties arise, the path parameter velocity shows almost no significant changes. Only at the end of the path, the path parameter automatically slows down, so that a smooth deceleration is achieved. The computing time for one control step is in the range of 40 ms.

In case 2 and 3, the obstacles shown in Fig. 4 are considered in the objective function or in both the objective function...
and the constraints respectively. The path parameter evolution in Fig. 5(b) shows that at the beginning for $t < 18$ s the path parameter increases slower and therefore the end-effector also moves slower along the path, as the robot moves towards the obstacles. When the robot passes the obstacles, the path parameter velocity increases and the robot moves as fast as without obstacles. The deviations from the path remain small.

VII. CONCLUSION

In this work, a Nonlinear Model Predictive Control approach for controlling the end-effector pose of redundant (mobile) manipulators has been presented. Additionally to the Cartesian task, collision avoidance in performed. Modeling the robot based on the Denavit-Hartenberg convention allows to apply the same control framework to both fixed and mobile manipulators. Simulation results of a 7 DoF manipulator and a 10 DoF mobile manipulator show the feasibility of the method.

Future work will integrate the control algorithm on real robots using the obstacle information obtained from workspace monitoring [18].

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