Whole-Body Kinematic Model Predictive Control for Continuous Mobile Manipulation Tasks

Johannes Pankert, Marco Hutter

***Abstract*— We present a receding horizon controller to perform continuous tasks with a mobile manipulator. The end effector of our robot follows a desired trajectory while respecting joint space limits and avoiding self-collisions. We use an airbrush to spray a pattern onto a wall to demonstrate the capabilities of the controller.**

I. Introduction

Many current works on mobile manipulation separate the task into a locomotion and a manipulation problem. We propose a whole-body control approach that allows for coordinated base-arm motion to solve the task. The controller enables our robot to operate in a workspace that is larger than the maximum reach of its manipulator. This contribution is an extension to our previous work, where we initially introduced our whole-body model predictive controller [1]. We add joint space constraints and propose a method for self-collision avoidance. The systems capabilities are demonstrated with an airbrush paint task on the mobile manipulator MabiMobile.

II. Method

A model predictive control (MPC) module generates control inputs for the robot to follow an end-effector trajectory while respecting several constraints. In the following sections, we describe the controller's system model and the cost function used.

A. System Model

We approximate the full system dynamics with a kinematic model in the MPC. Joint positions describe the state of the arm $\boldsymbol{x_{arm}} = [q_0, \dots q_5]^T$. In contrast to our previous work, the full base pose is used in the system model. A quaternion encodes the base's orientation and a $\Re^3$ vector its position. Together, they form the base state $\boldsymbol{x_{base}} = [\boldsymbol{q_{base}}, \boldsymbol{r}]^T$. For the state of the whole system, we write: $\boldsymbol{x} = [\boldsymbol{x_{base}}, \boldsymbol{x_{arm}}]^T$. Using the full base pose instead of a reduced state $[x, y, \varphi_{yaw}]$ for a ground robot has two advantages: 1. The robot can negotiate uneven terrain and track end effector references given in world frame accurately. 2. The robot used for evaluation of the method has a differential drive base with supporting passive castor wheels. These wheels are attached to the base with a spring suspension system. Depending on the arm configuration, the system's center of mass varies significantly and hence the compression status of the springs changes. This leads to a varying pitch angle of the base, even on flat terrain, which cannot be described with the reduced model.
The arm is controlled with joint velocity references $\boldsymbol{u_{arm}} = [\omega_0, \dots \omega_5]$. The base's wheel speeds are calculated from the desired base twist, the forward velocity and the turning rate $u_{base} = [v, \omega_b]$. The desired base twist can be applied to the base state in two ways: For most robotic system, applying a base frame twist to the robot state is a good choice. This is equivalent to approximating the ground with a plane perpendicular to the base normal.
Since our robot is mostly going to operate on flat grounds and has got an uncontrolled rotational pitch degree of freedom, we apply the control twist in world frame instead. Rotating around the world z-axis models reality better than rotating around the base normal vector.
These considerations lead to the following system model:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{q_{base}} \boxdot 0.5k\omega \\ \boldsymbol{q_{base}} \boxdot iv \boxdot \boldsymbol{q_{base}^{-1}} \\ \boldsymbol{u_{arm}} \end{bmatrix} \tag{1}$$

$\boxdot$ denotes the Hamilton quaternion product and i, j, k are the quaternion imaginary units.

B. Costfunction

The costfunction consists of multiple terms to balance the different objectives of the controller:

1) Task space tracking: Deviations of the robot's end effector pose $\boldsymbol{x_{ee}}$ to a desired pose $\hat{\boldsymbol{x}}_{\boldsymbol{ee}}$ are penalized. The desired pose may be time-dependent and varies over the controller's time horizon. For the translational error, we use the difference between the two position vectors. To compute the rotational error, we use an orientation error formulation on the orientation quaternions of the current and the desired pose [2].
The robot's end effector pose is a function of the current state. We use Robcogen [3] to compute the joint-angle-dependent transformation from the robot's base to the end effector and multiply with the world to base transform from $\boldsymbol{x_{base}}$. The sum of squares of these error functions form the end effector tracking term of our costfunction: $C_{ee}(t, \boldsymbol{x}) = \|\boldsymbol{x_{ee,pos}}(t, \boldsymbol{x}) - \hat{\boldsymbol{x}}_{\boldsymbol{ee,pos}}(t)\|_2^2 + \|e_O(\boldsymbol{x_{ee,orientation}}(t, \boldsymbol{x}), \hat{\boldsymbol{x}}_{\boldsymbol{ee,orientation}}(t)\|_2^2$.

2) Joint Space Constraints: In our previous work we used a quadratic penality cost to punish deviations from a nominal arm configuration. This prevented self-collision but hurt the accuracy of the end effector tracking objective. Here, we use Relaxed Barrier Functions (RBF) instead to enforce the joint limits of the robot [4].

$$\hat{B}(z) = \begin{cases} -ln(z) & z > \delta \\ \beta(z; \delta) & z <= \delta \end{cases} \tag{2}$$

β is the quadratic function for which $B \in \mathcal{C}^2$. The RBFs implement soft inequality constraints that penalize violations of the joint position and velocity limits. With $\delta = 0.1$ we get a good balance between constraint fulfillment and convergence of the SLQ solver.

3) Self-Collision avoidance: Collisions of arm and base are avoided by putting a cost on the end effector position relative to the robot's base frame. We define a differentiable cost function on the end effector planar position with RBFs. The cost pushes the end effector away from the base footprint. Another cost term prevents the end effector to move as far away as possible by penalizing its distance to the base's center. Figure 1a shows the combined end effector collision cost $C_{collision}$ with respect to the base origin plotted in logarithmic scale.

C. MPC Formulation

A weighted sum over the said cost terms is the intermediate MPC cost. An additional quadratic cost term penalizes control inputs.

$$J(\boldsymbol{x}, \hat{\boldsymbol{x}}, \boldsymbol{u}) = \int_{\tau=t_0}^{t_0+T} L(\boldsymbol{x}(\tau), \hat{\boldsymbol{x}}(\tau), \boldsymbol{u}(\tau)) d\tau \tag{3}$$

$$\begin{aligned} L(\boldsymbol{x}, \hat{\boldsymbol{x}}, \boldsymbol{u}) =& \lambda_0 C_{ee} + \lambda_1 C_{joint\_pos} + \lambda_2 C_{joint\_vel} + \\ & \lambda_3 C_{base\_collision} + \boldsymbol{u}^T \boldsymbol{R} \boldsymbol{u} \end{aligned} \tag{4}$$

The experiments were conducted with $\lambda_0 = 10$, $\lambda_{1,2,3} = 10^{-3}$ and $\boldsymbol{R} = diag[\mathbb{I}_2, \mathbb{I}_6]$. We implemented the MPC control problem with the OCS2 toolbox and used its SLQ solver [5] to compute optimal solutions with a rate of 100 Hz over a time horizon of $T = 2\,\mathrm{s}$. The necessary derivatives and Hessians were computed with CppAD Code Gen [1].

In contrast to our previous work, we do not follow the optimal trajectories with a separate tracking controller but we use the affine policies generated by the MPC. We evaluate the affine policies with the latest state estimates at a rate of 250 Hz and directly send the computed control inputs to the motor controllers.

III. Hardware Experiments

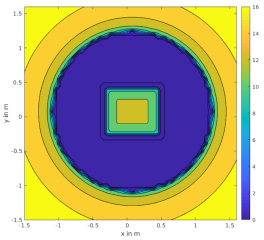
A. MabiMobile

In our previous work we implemented the whole-body model predictive controller on the Waco mobile manipulator [1]. In this work, we use the MabiMobile platform instead. MabiMobile consists of a differential drive base with supporting castor wheels and a Mabi Speedy 12 manipulator. The robot uses an Intel Realsense T265 for localization. Various tools can be attached to the end effector for different applications.

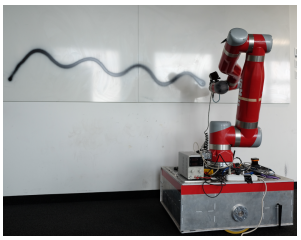
B. Experiment

We mounted a paint brush to the end effector and sprayed a pattern onto a whiteboard to showcase a continuous manipulation task. Figure 1b shows the drawn line. To solely evaluate the controller performance and disregard the problem of global localization, we command the spraying path relative to the initial end effector pose. This explains the vertical drift of the sprayed path.

[1] https://github.com/joaoleal/CppADCodeGen



(a) Cost on the end effector planar position in base frame. The optimal end effector position is outside the base footprint and within the maximum reach of the arm.



(b) MabiMobile spraying a black line onto a whiteboard.

$\mu_{pos}[\mathrm{m}]$	$\sigma_{pos}[\mathrm{m}]$	$\mu_{rot}[^\circ]$	$\sigma_{rot}[^\circ]$
0.0156	0.0035	2.3981	0.5039
0.0184	0.0111	2.3668	0.5848
0.0150	0.0036	2.4146	0.5038
0.0155	0.0031	2.4924	0.5349
0.0208	0.0042	3.1252	0.7177
0.0170	0.0039	2.7039	0.5628

TABLE I: Linear and angular deviations of the robot's end effector pose from a reference trajectory.

We measured the accuracy of the position and orientation tracking on 6 reference trajectories of 1.5 m length. Table I shows the mean positional and rotational deviations and their standard deviations of 200 recorded end effector poses from the reference trajectory.

IV. Conclusions

We presented a whole-body model predictive controller for continuous tasks. The MPC plans with the knowledge of joint limits and avoids self-collisions. Our proposed collision avoidance procedure can be extended to other task space obstacles.

We plan to use the controller to automize tasks in building construction such as plastering, grinding, or chiseling.

References

[1] A. Gawel, H. Blum, J. Pankert, K. Krämer, L. Bartolomei, and S. Ercan, "A Fully-Integrated Sensing and Control System for High-Accuracy Mobile Robotic Building Construction," *(accepted to) 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[2] B. Siciliano, Ed., *Robotics: modelling, planning and control*, ser. Advanced textbooks in control and signal processing. London: Springer, 2009, oCLC: ocn144222188.

[3] M. Frigerio, J. Buchli, D. G. Caldwell, and C. Semini, "{R}ob{C}o{G}en: a code generator for efficient kinematics and dynamics of articulated robots, based on {D}omain {S}pecific {L}anguages," vol. 7, no. Special Issue on Domain-Specific Languages and Models for Robotic Systems, pp. 36–54, 2016.

[4] C. Feller and C. Ebenbauer, "Relaxed Logarithmic Barrier Function Based Model Predictive Control of Linear Systems," *IEEE Transactions on Automatic Control*, 2017.

[5] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 93–100.