Aalborg Universitet

AALBORG UNIVERSITY

Path-Following Model Predictive Control of Ballbots

Jespersen, Thomas Kølbæk; Ahdab, Mohamad Al; Méndez, Juan de Dios Flores; Damgaard, Malte Rørmose; Hansen, Karl Damkjær; Pedersen, Rasmus; Bak, Thomas

Published in:
2020 IEEE International Conference on Robotics and Automation (ICRA)

DOI (link to publication from Publisher):
10.1109/ICRA40945.2020.9196634

Publication date:
2020

Document Version
Accepted author manuscript, peer reviewed version

Link to publication from Aalborg University

Citation for published version (APA):
Jespersen, T. K., Ahdab, M. A., Méndez, J. D. D. F., Damgaard, M. R., Hansen, K. D., Pedersen, R., & Bak, T. (2020). Path-Following Model Predictive Control of Ballbots. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1498-1504). Article 9196634 IEEE (Institute of Electrical and Electronics Engineers). https://doi.org/10.1109/ICRA40945.2020.9196634

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal -

Take down policy
If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from vbn.aau.dk on: July 03, 2025

Path-Following Model Predictive Control of Ballbots

Thomas K. Jespersen, Mohammad al Ahdab, Juan de Dios F. Mendez,
Malte R. Damgaard, Karl D. Hansen, Rasmus Pedersen and Thomas Bak

***Abstract*— This paper introduces a novel approach for model predictive control of ballbots for path-following tasks. Ballbots are dynamically unstable mobile robots which are designed to balance on a single ball. The model presented in this paper is a simplified version of a full quaternion-based model of ballbots' underactuated dynamics which is suited for online implementation. Furthermore, the approach is extended to handle nearby obstacles directly in the MPC formulation. The presented controller is validated through simulation on a high fidelity model as well as through real-world experiments on a physical ballbot system.**

I. Introduction

Safety is a very important element if robots should be able to navigate and solve tasks in human-occupied environments. These environments are challenging for robots because humans are dynamically moving obstacles and cluttered environments will leave only narrow passages for the robot to navigate. To accommodate the possible rapid changes in the scenes, planning and control algorithms must be able to produce and execute safe and feasible paths fast. Consequently, minimizing compute time is of interest.

Ballbots have unique holonomic properties and are well suited for cluttered environments and human-robot interaction. Not only can they move freely in any direction, but they also exhibit human-friendly motions as a result of their underactuated dynamics. Ballbots fall in the category of shape-accelerated systems [1], where the shape variables are the inclination and orientation of the ballbot, in the latter referred to as the attitude. Hence to accelerate ballbots have to lean. The planning and control algorithms need to consider these dynamics to ensure that the paths of interest can be executed.

A common approach to local planning for mobile robots is to use model predictive control. To make the controller computationally feasible in real-time some of the earliest and most popular approaches, such as the dynamic window approach (DWA) [2] and trajectory rollout [3], used simple dynamical models and reduced the search space of possible controls to constant translational and rotational velocities over a relatively short time horizon. These approaches are not suited for ballbots since they do not account for the underactuated dynamics while the constant control action greatly limits the useful horizon length. Besides, the advancements in computers over the last decade enables a larger search space of possible controls to be explored. Recently, this has been addressed by the tensor dynamic window approach (TDWA) [4], which uses a grid-map representation with an efficient collision checking approach. However, TDWA is limited to a fixed set of precomputed control sequences and also to the resolution of the grid-map.

For ballbots, past research have mainly focused on robust control of the attitude and velocity, without considering the dynamic motion behaviour that ballbots exhibit. The few that have considered dynamic motion planning include the gain-scheduled LQR controller proposed in [5] and similarly the iterated LQG scheme proposed in [6]. Focusing mainly on optimal motion, these two approaches do not consider obstacles and only considers point-to-point planning.

In [1] and [7], it is shown how shape trajectories can be computed through optimization given desired acceleration trajectories in position space. In [8] a differentially flat output of the ballbot is derived and used for trajectory optimization, enabling arbitrary motion and free constraints on e.g., the position. The proposed method is capable of converting an optimal kinematic path from a path planning algorithm to a dynamically feasible path for the ballbot, which converts into an angle trajectory.

This paper takes another approach and proves that the highly non-linear underactuated ballbot system can be controlled as a linear system which does not even include the underactuated properties. Given a path-following formulation we allow the final robot trajectory to deviate in time from the desired trajectory, hence allowing underactuated deviations without punishing them.

The approach presented in this paper uses an MPC formulation to control the motion of ballbots given a low-level balance controller. The focus of this work is to perform shape-space planning with an MPC to generate control actions that create progress along a planned path and avoid any obstacles at the expense of closely following the path.

The rationale is that the robot will be navigating corridors of large campuses like hospitals and airports, where progress through the corridor and sensible obstacle avoidance are prioritized. Furthermore, global planning is performed on high-level maps and simply presented as long straight lines in the middle of the corridors. This means that the robot should preferably progress naturally along the path, but not necessarily follow it closely. In contrast to DWA and Trajectory Rollout, this formulation allows varying control action along the horizon.

This work is supported by Innovation Fund Denmark in project number 7076-00051B called Robot Digital Signage.

T. K. Jespersen was with Department of Electronic Systems, Aalborg University, Denmark. He is now with APTIV (formerly nuTonomy), Singapore, thomasj@tkjelectronics.dk

M. al Ahdab, J. F. Mendez, M. R. Damgaard, K. D. Hansen, R. Pedersen and T. Bak are with Department of Electronic Systems, Aalborg University, Denmark, {mrd,maah,rpe,kdh,tba}@es.aau.dk

The MPC is designed on a simplified and linearized model without any underactuated dynamics but with a path-following cost function that compensates for the lack of underactuated dynamics. A set of lifted output constraints ensures that the MPC stays within the limits of the low-level controller and actuators. Obstacle avoidance is achieved by including nearby obstacles as both constraints and exponential cost.

The remainder of the paper is structured as follows; Sec. II outlines the ballbot system architecture. In Sec. III, a quaternion-based model of the ballbot is presented and simplified for use in the MPC. In Sec. IV, the shape-accelerated MPC approach is presented. Sec. V, validates the proposed control approach through both simulation and real-world experiments with a physical ballbot system. Finally, Sec. VI, concludes the work.

II. System Architecture

The system architecture generally follows that of mobile robots, where a user can input a goal, such as a desired location in a map. The goal is transformed into the desired path through a path planner utilizing feedback from SLAM and obstacle detection algorithms which depend on sensor feedback. Subsequently, the desired path, nearby obstacles, and velocity and heading estimates are forwarded to a path-following controller, which provides attitude references for a low-level balancing controller. The system architecture, along with a CAD model of the ballbot, is illustrated in Fig. 1. The computation effort is divided between a high-

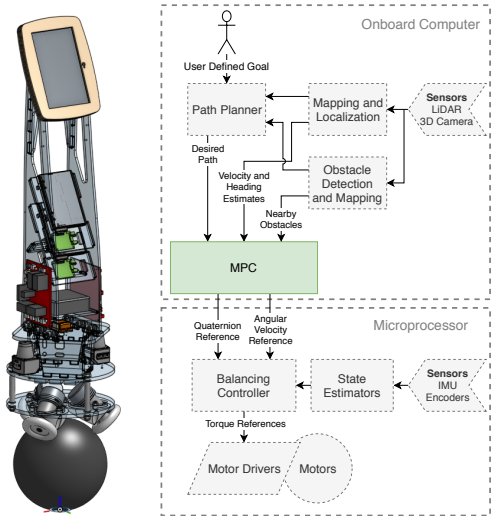

Fig. 1. 3D model of the considered ballbot along with a simplified sketch of the system architecture.

level onboard computer and a low-level real-time microcontroller. The onboard computer (Intel NUC) runs ROS, which handles SLAM and obstacle detection along with the path-following controller. The low-level microcontroller (ARM Cortex-M7) runs the balancing controller and local state estimators.

III. BallBot Model

Several approaches to modeling ball-balancing robots have been proposed in [5], [9]–[11]. Common to these models are either a decoupled planar model or an Euler-angle based coupled model resulting in complicated symbolic derivations, including an exhaustive mixture of sine and cosine expressions. This work takes a different approach by utilizing the quaternion-based model derived in [12] for a ballbot with three omni-wheels installed with 120° separation. The model

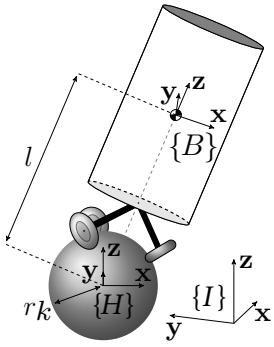

Fig. 2. Rigid body diagram of a ball balancing robot, with attached model frames. $\{I\}$ is the inertial frame, $\{H\}$ is the heading frame, and $\{B\}$ is the body frame.

decomposes the ballbot into two rigid bodies; one for the ball and one for the robot body, which balances on top of the ball as illustrated in Fig. 2. The model in Fig. 2 shows three frames: The inertial frame, $\{I\}$; the heading frame, $\{H\}$, which is always horizontal but aligned with the heading of the robot; and the body frame, $\{B\}$. The x-axis defines the forward direction of the robot and the heading angle is thus defined as the angle between the x-axis of the heading frame and the inertial frame. Moreover, three motors with omni-wheels are rigidly attached to the body. The wheels are assumed to roll on the ball without slip. The two rigid bodies are described by two degrees of freedom for the ball, which is assumed not to rotate around the vertical axis, and three degrees of freedom for the body. The generalized coordinates of the model are given by

$$\boldsymbol{\chi} = \begin{bmatrix} {}^{\mathrm{I}}x & {}^{\mathrm{I}}y & {}^{\mathrm{I}}_{\mathrm{B}}\boldsymbol{q} \end{bmatrix}^{\mathsf{T}} \tag{1}$$

where ${}^{\mathrm{I}}x$ and ${}^{\mathrm{I}}y$ are the inertial frame location of the center of the ball and ${}^{\mathrm{I}}_{\mathrm{B}}\boldsymbol{q}$ is the quaternion describing the attitude of the robot body in the inertial frame. The model assumes no slip between the ball and ground, such that the rotational degrees of freedom of the ball link directly to the translational movement of both the ball and the body. Symbolic forward and inverse kinematics models are derived for the odometry, enabling conversion between the angular velocities of the omni-wheels and the translational velocity of the ball given the current attitude of the body.

The dynamics of the ballbot are modelled using Lagrangian mechanics with the kinetic energy of the ball, T_k; the kinetic energy from rotational and translational movement of the body, T_b; the kinetic energy of the omni-wheels, T_w; and the potential energy of the body, V_b, which

stems from changes in height of the center of mass as the inclination changes. The Lagrangian is thus given as

$$\mathcal{L} = T_k + T_b + T_w - V_b \tag{2}$$

Due to the chosen dimension of the generalized coordinates in (1), six differential equations result from applying the Lagrangian to the Euler-Lagrange equation. Also, three friction terms are included to approximate the expected friction in the system: Viscous friction from ball to ground, $\boldsymbol{D}_K$; viscous friction from body angular velocity, $\boldsymbol{D}_B$; and viscous friction from motor angular velocity, $\boldsymbol{D}_M$, which is a combination of internal motor friction and ball to omni-wheel friction. These friction components are included in the equations of motion by adding the friction force matrix, $\boldsymbol{D}(\dot{\boldsymbol{\chi}})$.

$$\boldsymbol{D}(\dot{\boldsymbol{\chi}}) = \boldsymbol{D}_K(\dot{\boldsymbol{\chi}}) + \boldsymbol{D}_B(\dot{\boldsymbol{\chi}}) + \boldsymbol{D}_M(\dot{\boldsymbol{\chi}}) \tag{3}$$

The torque generated by the three omni-wheels, $\boldsymbol{\tau}_m$, acts in a coupled way on both the ball and the body. This input torque is included in the equations of motion by applying Hamilton's principle, where the inverse kinematic relationship is used to map the motor torques to the generalized coordinate space.

$$\boldsymbol{Q}(\boldsymbol{\chi}, \boldsymbol{\tau}_m) = \left(\frac{\partial \dot{\boldsymbol{\theta}}}{\partial \dot{\boldsymbol{\chi}}} \right)^{\mathsf{T}} \boldsymbol{\tau}_m \tag{4}$$

where $\dot{\boldsymbol{\theta}}$ (${}^{\mathrm{I}}_{\mathrm{B}}\boldsymbol{q}$, ${}^{\mathrm{I}}_{\mathrm{B}}\dot{\boldsymbol{q}}$, ${}^{\mathrm{I}}\dot{x}$, ${}^{\mathrm{I}}\dot{y}$) is the angular velocities of the three omni-wheels.

The dimension of the generalized coordinate space is, however, larger than the degrees of freedom of the system since all four quaternion elements are included. Thus, the holonomic quaternion constraint, ${}^{\mathrm{I}}_{\mathrm{B}}\boldsymbol{q}^{\mathsf{T}}\,{}^{\mathrm{I}}_{\mathrm{B}}\boldsymbol{q} = 1$, has to be included and eliminated using a Lagrange multiplier. The resulting equations of motion consists of six coupled differential equations in the form of

$$\widetilde{\boldsymbol{M}}(\boldsymbol{\chi})\,\ddot{\boldsymbol{\chi}} + \widetilde{\boldsymbol{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\,\dot{\boldsymbol{\chi}} + \widetilde{\boldsymbol{G}}(\boldsymbol{\chi}) + \widetilde{\boldsymbol{D}}(\dot{\boldsymbol{\chi}}) = \widetilde{\boldsymbol{Q}}(\boldsymbol{\chi})\,\boldsymbol{\tau}_m \tag{5}$$

where $\widetilde{\boldsymbol{M}}(\boldsymbol{\chi})$ is the mass/inertia matrix, $\widetilde{\boldsymbol{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ is the Coriolis and centrifugal force matrix, $\widetilde{\boldsymbol{G}}(\boldsymbol{\chi})$ is the gravitational force matrix, $\widetilde{\boldsymbol{D}}(\dot{\boldsymbol{\chi}})$ is the friction/dampening force matrix and $\widetilde{\boldsymbol{Q}}(\boldsymbol{\chi})$ is a transformation matrix that transforms the motor torques into equivalent torques in the generalized coordinate space. By reorganizing the equations the nonlinear quaternion-based control-affine model of the ballbot can be formulated as

$$\begin{aligned} \dot{\boldsymbol{x}} = &\begin{bmatrix} \dot{\boldsymbol{\chi}} \\ \widetilde{\boldsymbol{M}}(\boldsymbol{\chi})^{-1}\Big(-\widetilde{\boldsymbol{C}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\,\dot{\boldsymbol{\chi}} - \widetilde{\boldsymbol{G}}(\boldsymbol{\chi}) - \widetilde{\boldsymbol{D}}(\dot{\boldsymbol{\chi}})\Big) \end{bmatrix} \\ &+ \begin{bmatrix} \boldsymbol{0}_{6\times 3} \\ \widetilde{\boldsymbol{M}}(\boldsymbol{\chi})^{-1}\,\widetilde{\boldsymbol{Q}}(\boldsymbol{\chi}) \end{bmatrix} \boldsymbol{\tau}_m \end{aligned} \tag{6}$$

where the state vector is the concatenation of the generalized coordinates and their derivatives i.e. $\boldsymbol{x} = [\boldsymbol{\chi}; \dot{\boldsymbol{\chi}}]$, resulting in

$$\boldsymbol{x} = \begin{bmatrix} {}^{\mathrm{I}}x & {}^{\mathrm{I}}y & {}^{\mathrm{I}}_{\mathrm{B}}\boldsymbol{q}^{\mathsf{T}} & {}^{\mathrm{I}}\dot{x} & {}^{\mathrm{I}}\dot{y} & {}^{\mathrm{I}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} \tag{7}$$

Note that the rank of the input matrix is smaller than the degrees of freedom of the system resulting in the underactuated dynamics. For more details on the quaternion model consult [12].

A. Simplification of Model for MPC Implementation

The attitude dynamics of a ballbot in a region close to the upright position can be assumed sufficiently faster than the translational dynamics [12] such that it is possible to design two controllers in a cascaded configuration. A nonlinear sliding mode controller is designed in [12] to stabilize the attitude of the body to a given body reference frame, $\{\tilde{B}\}$, given a quaternion reference, ${}^{\mathrm{I}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}$, and an angular velocity reference, ${}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}$. The evolution of the quaternion reference is assumed to be kinematically linked to the angular velocity reference. Since the sliding mode controller is designed to be robust against matched uncertainties, it can be assumed that the balance controller will stabilize the attitude while driving in any direction with any velocity within the operational envelope. This envelope is limited by the dynamics of the balance controller which is closely related to the limitations of motor torque and friction, to avoid slip. Designing an MPC for the translational dynamics will thus only require a subset of the full quaternion-based model from Sec. III. However, the translational subset is still coupled with the full dynamics making it infeasible for real-time solving.

By assuming a robust and fast balance controller, a MPC can be designed using the underactuated shape-acceleration relationship between the body inclination and translational acceleration. A linear shape-space accelerated model is derived in (8) by linearizing a heading-independent version of the quaternion-based model from (6), with the inclusion of the closed loop balance controller. The heading direction is defined as the 2D projection of the body x-axis vector onto the xy-plane of the inertial frame. The attitude quaternion defined in the heading frame furthermore captures the decoupled inclination in the x and y elements of the quaternion such that the acceleration coefficients can be computed from

$$\boldsymbol{A}_{\dot{\boldsymbol{v}}\boldsymbol{q}} = \left[\begin{matrix} \frac{\partial\, {}^{\mathrm{H}}\ddot{x}}{\partial\, {}^{\mathrm{H}}_{\mathrm{B}}\boldsymbol{q}} \\ \frac{\partial\, {}^{\mathrm{H}}\ddot{y}}{\partial\, {}^{\mathrm{H}}_{\mathrm{B}}\boldsymbol{q}} \end{matrix} \right]\Bigg|_{\substack{\boldsymbol{x}=\tilde{\boldsymbol{x}} \\ \boldsymbol{q}_{\mathrm{ref}}=\tilde{\boldsymbol{q}}_{\mathrm{ref}} \\ \boldsymbol{\omega}_{\mathrm{ref}}=\tilde{\boldsymbol{\omega}}_{\mathrm{ref}}}}, \quad \boldsymbol{A}_{\dot{\boldsymbol{v}}\boldsymbol{q}_r} = \left[\begin{matrix} \frac{\partial\, {}^{\mathrm{H}}\ddot{x}}{\partial\, {}^{\mathrm{H}}_{\mathrm{B}}\boldsymbol{q}_{\mathrm{ref}}} \\ \frac{\partial\, {}^{\mathrm{H}}\ddot{y}}{\partial\, {}^{\mathrm{H}}_{\mathrm{B}}\boldsymbol{q}_{\mathrm{ref}}} \end{matrix} \right]\Bigg|_{\substack{\boldsymbol{x}=\tilde{\boldsymbol{x}} \\ \boldsymbol{q}_{\mathrm{ref}}=\tilde{\boldsymbol{q}}_{\mathrm{ref}} \\ \boldsymbol{\omega}_{\mathrm{ref}}=\tilde{\boldsymbol{\omega}}_{\mathrm{ref}}}} \tag{8}$$

where ${}^{\mathrm{H}}\ddot{x}$ and ${}^{\mathrm{H}}\ddot{y}$ defines the acceleration of the ballbot in the heading frame, and ${}^{\mathrm{H}}_{\mathrm{B}}\boldsymbol{q}$ and ${}^{\mathrm{H}}_{\mathrm{B}}\boldsymbol{q}_{\mathrm{ref}}$ are the attitude quaternion and quaternion reference defined in the heading frame. The model is linearized around the operating point defined in (9), being the upright equilibrium where the balance controller tracks the zero inclination reference with zero angular velocity reference and zero translational velocity.

$$\begin{aligned} &\tilde{\boldsymbol{q}} = \tilde{\boldsymbol{q}}_{\mathrm{ref}} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^{\mathsf{T}},\ \dot{\tilde{\boldsymbol{q}}} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^{\mathsf{T}}, \\ &\tilde{\boldsymbol{\omega}}_{\mathrm{ref}} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^{\mathsf{T}},\ {}^{\mathrm{I}}\tilde{\dot{x}} = 0,\ {}^{\mathrm{I}}\tilde{\dot{y}} = 0 \end{aligned} \tag{9}$$

The linearized model matrices $\boldsymbol{A}_{\dot{\boldsymbol{v}}\boldsymbol{q}}$ and $\boldsymbol{A}_{\dot{\boldsymbol{v}}\boldsymbol{q}_r}$ captures the linearized relationship between the inclination-related

elements of the two attitude quaternions and the resulting acceleration in the heading frame [12]. The relationship is shown in (10) with the two coefficients, c_{qx} and c_{qy}, assuming that the balance controller tracks the quaternion reference perfectly.

$$\begin{bmatrix} {}^{\mathrm{H}}\ddot{x} \\ {}^{\mathrm{H}}\ddot{y} \end{bmatrix} = (\boldsymbol{A}_{\dot{\boldsymbol{v}}\boldsymbol{q}} + \boldsymbol{A}_{\dot{\boldsymbol{v}}\boldsymbol{q}_r}) \begin{bmatrix} {}^{\mathrm{H}}_{\mathrm{B}}q_1 \\ {}^{\mathrm{H}}_{\mathrm{B}}q_2 \end{bmatrix} = \begin{bmatrix} 0 & c_{qx} \\ -c_{qy} & 0 \end{bmatrix} \begin{bmatrix} {}^{\mathrm{H}}_{\mathrm{B}}q_1 \\ {}^{\mathrm{H}}_{\mathrm{B}}q_2 \end{bmatrix} \tag{10}$$

where ${}^{\mathrm{H}}_{\mathrm{B}}q_1$ and ${}^{\mathrm{H}}_{\mathrm{B}}q_2$ are the x and y elements of the attitude quaternion defined in the heading frame, respectively. The off-diagonal terms are zero as a result of the linearization point with no inclination and an assumed perfectly aligned center of mass [12].

The optimization problem within the MPC is thus subject to the following simplified shape-accelerated dynamics

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}t}{}^{\mathrm{H}}_{\mathrm{B}}q_1 = \frac{1}{2}\,{}^{\tilde{\mathrm{B}}}\omega_{\text{ref,x}}, && \frac{\mathrm{d}}{\mathrm{d}t}s = \dot{s} && \\ &\frac{\mathrm{d}}{\mathrm{d}t}{}^{\mathrm{H}}_{\mathrm{B}}q_2 = \frac{1}{2}\,{}^{\tilde{\mathrm{B}}}\omega_{\text{ref,y}}, && \frac{\mathrm{d}}{\mathrm{d}t}\dot{s} = \ddot{s} && \frac{\mathrm{d}}{\mathrm{d}t}\,{}^{\mathrm{H}}\dot{x} = c_{qx}\,{}^{\mathrm{H}}_{\mathrm{B}}q_2, \\ &\frac{\mathrm{d}}{\mathrm{d}t}\,{}^{\mathrm{H}}x = {}^{\mathrm{H}}\dot{x}, && \frac{\mathrm{d}}{\mathrm{d}t}\,{}^{\tilde{\mathrm{B}}}\omega_{\text{ref,x}} = {}^{\tilde{\mathrm{B}}}\dot{\omega}_{\text{ref,x}} && \frac{\mathrm{d}}{\mathrm{d}t}\,{}^{\mathrm{H}}\dot{y} = -c_{qy}\,{}^{\mathrm{H}}_{\mathrm{B}}q_1, \\ &\frac{\mathrm{d}}{\mathrm{d}t}\,{}^{\mathrm{H}}y = {}^{\mathrm{H}}\dot{y}, && \frac{\mathrm{d}}{\mathrm{d}t}\,{}^{\tilde{\mathrm{B}}}\omega_{\text{ref,y}} = {}^{\tilde{\mathrm{B}}}\dot{\omega}_{\text{ref,y}} && \end{aligned} \tag{11}$$

where s is the path parametrization variable capturing progress along the path and optimized online using the corresponding control variable $\ddot{s}$. The MPC problem has been lifted such as to define ${}^{\tilde{\mathrm{B}}}\dot{\omega}_{\text{ref,x}}$ and ${}^{\tilde{\mathrm{B}}}\dot{\omega}_{\text{ref,y}}$ as the control variables of the MPC instead of ${}^{\tilde{\mathrm{B}}}\omega_{\text{ref,x}}$ and ${}^{\tilde{\mathrm{B}}}\omega_{\text{ref,y}}$.

IV. Controller Synthesis

The objective of the shape-accelerated MPC is to follow a reference path given by a high-level path planner as a connected series of geometric reference points, by computing an optimal sequence of balance controller references, ${}^{\tilde{\mathrm{B}}}\omega_{\text{ref,x}}$ and ${}^{\tilde{\mathrm{B}}}\omega_{\text{ref,y}}$, over a horizon of N samples using the model in (11).

A. Trajectory Tracking vs. Path-following

The MPC optimization problem is formulated as a path following problem given a reference path. In contrast to trajectory tracking, the reference points are given as a parametrized geometric path instead of a time-series of reference points. This adds time as an extra controllable degree of freedom [13] enabling the MPC to shape the acceleration based on the cost function rather than punishing deviations from a time-discretized reference trajectory, which are likely to happen due to the underactuated dynamics. Since the simplified shape-accelerated dynamics presented in (11) does not capture the underactuated dynamics of the ballbot, e.g., how the ballbot initially has to drive backward to change its inclination to accelerate forward, the extra degree of freedom in time will make the controller less susceptible to this lack of dynamics.

B. Path Parametrization

The MPC creates a local path from the global reference trajectory, as an arc curve length parametrized polynomial of order n_p. The polynomial order should be high enough to ensure continuous differentiation in position, velocity, and acceleration, resulting in a smooth trajectory. However, there are no closed-form expressions for the arc curve length of a polynomial of arbitrary degree [14], but it can be approximated by several closed-form expressions as shown in [15]–[17].

A sequence of local reference points are extracted and the Euclidean distance between the points serves as a first approximation of the arc curve length. The extracted reference points are fitted to a polynomial parametrized by this Euclidean distance. Finally the polynomial is refined by computing the approximate arc curve length using the closed-form expression from [16] and refitting a new polynomial to regularly spaced points extracted from the first polynomial. The resulting parametrized reference path is given as

$${}^{\mathrm{H}}x_{\text{ref}}(s) = \sum_{i=0}^{n_p} c_{\text{x,i}}\, s^i, \qquad {}^{\mathrm{H}}y_{\text{ref}}(s) = \sum_{i=0}^{n_p} c_{\text{y,i}}\, s^i \tag{12}$$

where $c_{\text{x,i}}$ and $c_{\text{y,i}}$ for $i = 0, \ldots, n_p$ defines the polynomial coefficients. Other common parametrizations such as Bezier curves [18], splines [19], and the Dubins path [20] can also be considered. Other examples of path-following MPC can be found in [21]–[23].

C. Obstacle Handling

Obstacles are included as position constraints with a circular keep-out area defined by a center and a radius $\{{}^{\mathrm{H}}\boldsymbol{O}_i,\, r_i\}$. The immediate distance to an obstacle, $p_{\text{o,i}}$, is computed as

$$p_{\text{o,i}} = \sqrt{({}^{\mathrm{H}}x - {}^{\mathrm{H}}x_{\text{o,i}})^2 + ({}^{\mathrm{H}}y - {}^{\mathrm{H}}y_{\text{o,i}})^2} - r_i - r_b \tag{13}$$

where $r_b = 0.1$ m is the radius of the robot and ${}^{\mathrm{H}}x_{\text{o,i}}$ and ${}^{\mathrm{H}}y_{\text{o,i}}$ defines the center of the obstacle, ${}^{\mathrm{H}}\boldsymbol{O}_i$. The MPC handles the obstacles by imposing the following constraints

$$p_{\text{o,i}} \geq -\gamma_o \quad \text{for } i = 1, \ldots, n_o \quad \text{and} \quad \gamma_o \geq 0 \tag{14}$$

where n_o defines an upper limit for the number of nearby obstacles to be considered by the MPC to make the optimization problem computationally feasible. At each control iteration the n_o number of nearest obstacles are thus extracted and used as constraints. Imposing only constraints to handle obstacle avoidance is dangerous since the MPC will end up driving on the constraint boundary. In the case of noise, dynamic variations, etc., the constraints are likely to be voided, resulting in an infeasible optimization problem. An obstacle avoidance cost similar to a barrier function is added to the optimization problem to push the robot away from the constraint boundary, thereby keeping a distance to the obstacles. The obstacle avoidance cost is defined as

$$e_{\text{obs}} = \sum_{i=1}^{n_o} \mathrm{e}^{k_p(-p_{\text{o,i}} + c_p)} \tag{15}$$

where k_p and c_p are the gain and offset values respectively, that defines the exponential barrier function used to push the robot away from the obstacles. This formulation is efficient for static obstacles and slow dynamic obstacles due to the feedback from sensors and constant re-computation of the trajectory.

D. Optimization Problem

The state vector of the MPC is defined as

$$\boldsymbol{X} = \begin{bmatrix} {}^{\text{H}}_{\text{B}}q_1 & {}^{\text{H}}_{\text{B}}q_2 & {}^{\text{H}}x & {}^{\text{H}}y & {}^{\text{H}}\dot{x} & {}^{\text{H}}\dot{y} & s & \dot{s} & {}^{\tilde{\text{B}}}\omega_{\text{ref,x}} & {}^{\tilde{\text{B}}}\omega_{\text{ref,y}} \end{bmatrix}^{\mathsf{T}} \tag{16}$$

To avoid infeasibility, slack variables are added to allow slight constraint violations at a high cost. This results in the following control variables within the MPC

$$\boldsymbol{u} = \begin{bmatrix} {}^{\tilde{\text{B}}}\dot{\omega}_{\text{ref,x}} & {}^{\tilde{\text{B}}}\dot{\omega}_{\text{ref,y}} & \ddot{s} & \gamma_v & \gamma_q & \gamma_o \end{bmatrix}^{\mathsf{T}} \tag{17}$$

where γ_v, γ_q and γ_o are slack variables used in the constraints for the velocity bounds, quaternion bounds, and obstacles avoidance respectively. Note how the control outputs have been lifted to angular acceleration, ${}^{\tilde{\text{B}}}\dot{\omega}_{\text{ref,x}}$, ${}^{\tilde{\text{B}}}\dot{\omega}_{\text{ref,y}}$, and path acceleration, $\ddot{s}$, such that acceleration constraints, which are closely related to motor torques, can be handled to keep the torques below saturation. The MPC minimizes the least squares cost defined in (18) combining a set of error terms, state variables and control variables over the horizon.

$$J = \boldsymbol{h}_N^{\mathsf{T}}\, \boldsymbol{W}_N\, \boldsymbol{h}_N + \sum_{k=0}^{N-1} \boldsymbol{h}_k^{\mathsf{T}}\, \boldsymbol{W}\, \boldsymbol{h}_k \tag{18}$$

where the cost vectors evaluated at the different time instances over the horizon is defined as

$$\boldsymbol{h}_k = \begin{bmatrix} \boldsymbol{e}_c & \boldsymbol{q}_{1,2} & \boldsymbol{\omega}_{\text{ref}} & \dot{\boldsymbol{\omega}}_{\text{ref}} & \gamma_v & \gamma_q & \gamma_o \end{bmatrix}^{\mathsf{T}} \tag{19}$$

$$\boldsymbol{h}_N = \begin{bmatrix} \boldsymbol{e}_c & \boldsymbol{q}_{1,2} & \boldsymbol{\omega}_{\text{ref}} \end{bmatrix}^{\mathsf{T}} \tag{20}$$

with

$$\begin{aligned}
&\boldsymbol{e}_c = \begin{bmatrix} e_{\text{lon}} & e_{\text{lat}} & e_{\text{vel}} & e_{\text{prog}} & e_{\text{obs}} \end{bmatrix}, \quad \begin{bmatrix} e_{\text{lon}} \\ e_{\text{lat}} \end{bmatrix} = \boldsymbol{R}_z^{\psi_{\text{ref}}}\, \boldsymbol{p}_{\text{err}} \\
&e_{\text{vel}} = v_{\text{lon}} - v_{\text{ref}}, && e_{\text{prog}} = s - s_{\max} \\
&\boldsymbol{q}_{1,2} = \begin{bmatrix} {}^{\text{H}}_{\text{B}}q_1 & {}^{\text{H}}_{\text{B}}q_2 \end{bmatrix}, && \boldsymbol{\omega}_{\text{ref}} = \begin{bmatrix} {}^{\tilde{\text{B}}}\omega_{\text{ref,x}} & {}^{\tilde{\text{B}}}\omega_{\text{ref,y}} \end{bmatrix} \\
&v_{\text{lon}} = \begin{bmatrix} {}^{\text{H}}\dot{x} & {}^{\text{H}}\dot{y} \end{bmatrix} \cdot \boldsymbol{d}_{\text{ref}}, && \dot{\boldsymbol{\omega}}_{\text{ref}} = \begin{bmatrix} {}^{\tilde{\text{B}}}\dot{\omega}_{\text{ref,x}} & {}^{\tilde{\text{B}}}\dot{\omega}_{\text{ref,y}} \end{bmatrix} \\
&\boldsymbol{d}_{\text{ref}} = \begin{bmatrix} \cos(\psi_{\text{ref}}) \\ \sin(\psi_{\text{ref}}) \end{bmatrix}, && \boldsymbol{p}_{err} = \begin{bmatrix} {}^{\text{H}}x - {}^{\text{H}}x_{\text{ref}}(s) \\ {}^{\text{H}}y - {}^{\text{H}}y_{\text{ref}}(s) \end{bmatrix} \\
&\psi_{\text{ref}} = \text{atan2}({}^{\text{H}}\dot{y}_{\text{ref}}(s), {}^{\text{H}}\dot{x}_{\text{ref}}(s))
\end{aligned} \tag{21}$$

where e_{lon} is the longitudinal error, e_{lat} is the lateral error, e_{vel} is the velocity error, e_{prog} is the progress error, $\boldsymbol{p}_{\text{err}}$ is the position error to the closest point on the path, $\boldsymbol{R}_z^{\psi_{\text{ref}}}$ is a clockwise rotating 2D rotation matrix of the heading reference, v_{lon} is the longitudinal velocity and $\boldsymbol{d}_{\text{ref}}$ is the direction vector of the path at a given point. The linear velocity reference $\{{}^{\text{H}}\dot{x}_{\text{ref}},\ {}^{\text{H}}\dot{y}_{\text{ref}}\}$ is computed from differentiation of the reference path polynomial. The error on the progress, e_{prog}, will ensure the progress on the path although it will affect the resulting velocity if the weight is in the same magnitude as the weight on the velocity error.

$$e_{\text{end}} = s - s_{\max} \tag{22}$$

Another type of progress error, e_{end}, pushes the system towards the end of the fitted trajectory by rewarding the driven distance related to the path parameter, s. Lastly, the constraints are defined as

$$\begin{aligned}
0 &\le s &&\le s_{\max}, \\
-q_{\lim} &\le {}^{\text{H}}_{\text{B}}q_1 &&\le q_{\lim}, \\
-q_{\lim} &\le {}^{\text{H}}_{\text{B}}q_2 &&\le q_{\lim}, \\
-\omega_{\lim} &\le {}^{\tilde{\text{B}}}\omega_{\text{ref,x}} &&\le \omega_{\lim}, \\
-\omega_{\lim} &\le {}^{\tilde{\text{B}}}\omega_{\text{ref,y}} &&\le \omega_{\lim}, \\
-\dot{\omega}_{\lim} &\le {}^{\tilde{\text{B}}}\dot{\omega}_{\text{ref,x}} &&\le \dot{\omega}_{\lim}, \\
-\dot{\omega}_{\lim} &\le {}^{\tilde{\text{B}}}\dot{\omega}_{\text{ref,y}} &&\le \dot{\omega}_{\lim},
\end{aligned} \qquad \begin{aligned}
\dot{s} &\ge 0 \\
v &\le v_{\max} + \gamma_v \\
p_{\text{o,i}} &\ge -\gamma_o \\
\gamma_v &\ge 0 \\
\gamma_q &\ge 0 \\
\gamma_o &\ge 0
\end{aligned} \tag{23}$$

where v is the Euclidean norm of the velocity of the robot.

The maximum quaternion constraint helps to limit the maximum inclination to an angle of $\theta_{\lim}$ according to

$$q_{\lim} = \sin(0.5\,\theta_{\lim}) + \gamma_q \tag{24}$$

which in return can be related to a maximum acceleration.

To ensure stability and feasibility terminal constraints are included. By ensuring that the robot has an upright inclination at the end of the horizon, the terminal acceleration is zero. Therefore, the quaternion elements ${}^{\text{H}}_{\text{B}}q$ and ${}^{\text{H}}_{\text{B}}q$ should both be zero at the end of the horizon together with the angular velocity references ${}^{\tilde{\text{B}}}\omega_{\text{ref,x}}[N]$ and ${}^{\tilde{\text{B}}}\omega_{\text{ref,y}}[N]$ so that the quaternion elements remain at zero. The velocity norm, $v[N]$, should be kept below the maximum to allow only slack within the horizon. Finally, the travelled distance, $s[N]$, is limited to a valid region of the fitted reference polynomial as any reference beyond that point would be unmodelled.

$$\begin{aligned}
&{}^{\text{H}}_{\text{B}}q_1[N] = 0 && {}^{\text{H}}_{\text{B}}q_2[N] = 0 && {}^{\tilde{\text{B}}}\omega_{\text{ref,x}}[N] = 0 \\
&{}^{\tilde{\text{B}}}\omega_{\text{ref,y}}[N] = 0 && v[N] \le v_{\max} && s[N] \le s_{\max}
\end{aligned} \tag{25}$$

The weighting matrices $\boldsymbol{W}$ and $\boldsymbol{W}_N$ are chosen to be diagonal with the diagonal elements representing the individual weights for cost term.

V. Test and Validation

The proposed path-following MPC has been evaluated in simulation and experimentally tested on a real robot. For test and verification of the balance controller the reader is referred to [12]. The simulation was carried out in MATLAB Simulink 2018b.

A. Test Scenario

The robot is given a circular trajectory centred at the origin of the inertial frame and with a radius of 1 m while keeping the initial heading constant and maintaining a desired longitudinal velocity of 0.25 m/s. Furthermore, four circular obstacles are included. The final MPC parameters and the defined obstacles can be found in Table I and II in

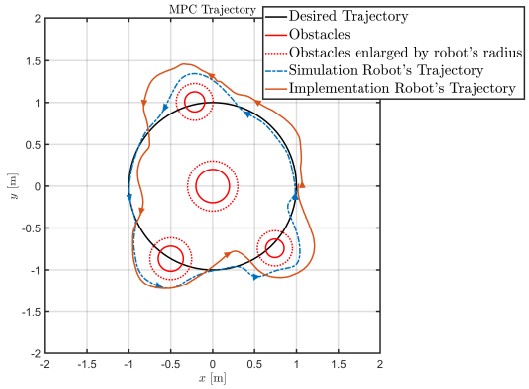

Fig. 3. Trajectory tracking for both simulation and test. The arrows indicate the progression of the robot along the trajectory. The inflated obstacles have been enlarged with the robot's radius.

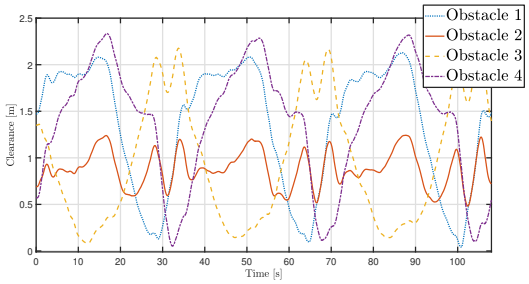

Fig. 4. Clearance between the robot and all the obstacles in the experimental run.

the Appendix. The MPC is developed as a ROS package which communicates with the embedded microcontroller on the robot. For mapping and localization the gmapping package [24] is used with two single beam SICK LiDAR sensors on the robot. The MPC was run at a sampling frequency of 10 Hz on an external computer (Intel Core i7-6600U, 8 GB RAM) connected to the shared ROS network over WiFi. The code used for simulation and implementation can be found at [25] and [26] respectively.

B. Results

The results of the path following experiments are captured in Fig. 3, along with simulation results for the same scenario. Furthermore, Fig. 4 shows the clearance of the robot to the obstacles and Fig. 5 illustrates the cost function values and velocity norm for both the simulation and the test.

The robot successfully manages to avoid the obstacles while progressing along the desired circular path. However, there are slight differences between the simulation and experimental results. As expected the simulated trajectory of the robot shows a better tracking performance. Moreover, it can be noticed from Fig. 5 that the cost functions matches in behaviour for both of the cases with three consecutive spikes representing the situations where the robot gets close to one of the three obstacles along the desired path. The shift in time can be explained by the velocity plot which shows that the robot drives slower near the obstacles during the real-world test than in simulation. Besides, there is a

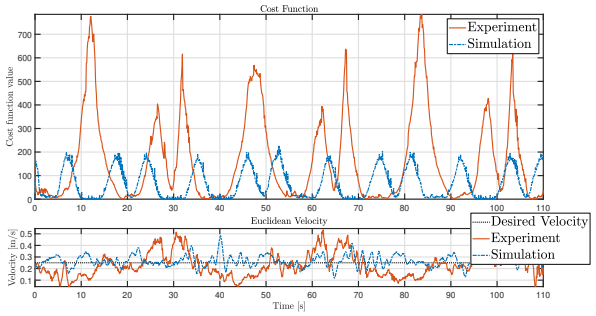

Fig. 5. Cost function values for both the test and simulation cases, together with the velocity norm of the robot.

difference in the magnitude of the cost functions especially at the spikes. These differences are likely due to model inaccuracies and/or uncertainties. For instance, the centre of mass of the actual robot is slightly offset compared to the nominal model used in simulation. These issues could be reduced by e.g., considering robust techniques for the MPC or online parameter estimation and compensation.

VI. Conclusion

This paper presented a novel path-following and obstacle avoidance model predictive control approach for ball balancing robots with underactuated dynamics. The MPC utilized a simplified quaternion-based model of the ballbot which made real-time implementation and computation feasible although orientation singularities are generally not a problem with ballbots [12]. The MPC was validated through simulation and experiments on a real ballbot to verify the approach. Although there were discrepancies between the simulation and the experiment, where the controller showed better tracking performance in simulation, these differences were as expected given the possible uncertainties such as unmodelled dynamics and inaccurate model parameters. Therefore, further testing and tuning of the MPC in real-life scenarios including external disturbances would be a natural direction for future work.

Appendix

TABLE I
MPC PARAMETERS FOR THE SIMULATION AND TEST

$W_{\text{lon}} = 20000$	$W_{\text{lat}} = 75$	$W_v = 600$	$W_{\text{prog}} = 0$	$W_q = 20$
$W_\omega = 10$	$W_{\dot{\omega}} = 20$	$W_{\gamma_v} = 9999$	$W_{\gamma_q} = 9999$	$W_{\gamma_o} = 9999$
$W_{\text{obs}} = 10$	$k_p = 8$	$c_p = 0.15$	$N = 22$	$f = 10\,\text{Hz}$
$\theta_{\text{lim}} = 7^\circ$	$\omega_{\text{lim}} = 10^\circ/\text{s}$	$\dot{\omega}_{\text{lim}} = 7^\circ/\text{s}^2$	$\nu_{\text{min}} = 0\,\text{m/s}$	$\nu_{\text{max}} = 3\,\text{m/s}$
$n_p = 8$	$n_o = 4$			

TABLE II
OBSTACLE PARAMETERS USED FOR SIMULATION AND TEST

Obstacle number	Center [m]	Radius [m]
1	$(-0.5, -\sqrt{3}/2)$	0.15
2	(0,0)	0.2
3	(−0.2141,1.0075)	0.12
4	$(1.05/\sqrt{2}, -1.05/\sqrt{2})$	0.11

REFERENCES

[1] U. Nagarajan, "Dynamic Constraint-based Optimal Shape Trajectory Planner for Shape-Accelerated Underactuated Balancing Systems," in *Robotics: Science and Systems 2010*, 2010, p. 8.

[2] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[3] B. Gerkey and K. Konolige, "Planning and control in unstructured terrain," in *ICRA Workshop on Path Planning on Costmaps*, 2008.

[4] O. Y. Sinyavskiy, J.-B. Passot, and B. Ibarz Gabardos, "Parallel Algorithm for Precise Navigation Using Black-Box Forward Model and Motion Primitives," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2423–2430, jul 2019.

[5] C. Gwerder and P. Fankhauser, "Modeling and Control of a Ballbot," Ph.D. dissertation, ETH Zürich, 2010.

[6] M. Neunert, F. Farshidian, and J. Buchli, "Adaptive Real-time Nonlinear Model Predictive Motion Control," 2014, p. 6.

[7] U. Nagarajan and R. Hollis, "Shape space planner for shape-accelerated balancing mobile robots," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1323–1341, Sep. 2013.

[8] M. Shomin, "Navigation and Physical Interaction with Balancing Robots," PhD Thesis, Carnegie Mellon University, Oct. 2016, cMU-RI-TR-16-58.

[9] T. Lauwers, G. Kantor, and R. Hollis, "A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* Orlando, FL, USA: IEEE, 2006, pp. 2884–2889.

[10] M. Kumagai and T. Ochiai, "Development of a robot balancing on a ball," in *2008 International Conference on Control, Automation and Systems*. Seoul, South Korea: IEEE, Oct. 2008, pp. 433–438.

[11] P. Dinh Ba, K. Hyung, K. Jaejun, and L. Soon-Geul, "Balancing and Transferring Control of a Ball Segway Using a Double-Loop Approach," *IEEE Control Systems*, vol. 38, no. 2, pp. 15–37, Apr. 2018.

[12] T. K. Jespersen, "Kugle - Modelling and Control of a Ball-balancing robot," Master Thesis, Aalborg University, Aalborg, Denmark, Apr. 2019.

[13] T. Faulwasser, "Optimization-based Solutions to Constrained Trajectory-tracking and Path-following Problems," PhD Thesis, Otto-von-Guericke-Universität Magdeburg, Oct. 2012.

[14] W. College, "Arc length and curvature."

[15] M. Madi, "Closed-form expressions for approximation of arclength parameterization for bézier curves," *In Int. J. Appl. Math. Comput. Sci*, pp. 33–41, 2004.

[16] M. Walter and A. Fournier, "Approximate arc length parametrization," p. 8, 1996.

[17] H. Wang, J. Kearney, and K. Atkinson, "Arc-Length Parameterized Spline Curves for Real-Time Simulation," *Curve and Surface Design*, vol. Saint-Malo 2002, pp. 387–396, iSBN 0-9728482-0-7.

[18] M. S. Floater and T. Surazhsky, "Parameterization for Curve Interpolation," in *Studies in Computational Mathematics*. Elsevier, 2006, vol. 12, pp. 39–54.

[19] K. Atkinson, "Modelling a Road Using Spline Interpolation," The University of Iowa, Tech. Rep., Feb. 2002.

[20] S. M. , "Dubins Curves," 2006.

[21] A. Liniger, A. Domahidi, and M. Morari, "Optimization-Based Autonomous Racing of 1:43 Scale RC Cars," *arXiv e-prints*, p. arXiv:1711.07300, Nov 2017.

[22] R. Frezza, A. Beghi, and A. Saccon, "Model predictive for path following with motorcycles: application to the development of the pilot model for virtual prototyping," in *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, vol. 1, Dec 2004, pp. 767–772 Vol.1.

[23] M. Hauan Arbo, E. Ingar Grøtli, and J. T. Gravdahl, "On Model Predictive Path Following and Trajectory Tracking for Industrial Robots," *arXiv e-prints*, p. arXiv:1703.02279, Mar 2017.

[24] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, Feb 2007.

[25] "Kugle ROS - repo for ROS related code," https://github.com/mindThomas/Kugle-ROS.

[26] "Kugle MATLAB - repo for MATLAB related code," https://github.com/mindThomas/Kugle-MATLAB.