Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 50-1 (2017) 531–536

OPTIPLAN: A Matlab Toolbox for Model Predictive Control with Obstacle Avoidance

Filip Janeček * **Martin Klaučo** * **Martin Kalúz** * **Michal Kvasnica** *

* *Slovak University of Technology in Bratislava, Slovakia (e-mail: {filip.janecek, martin.klauco, martin.kaluz, michal.kvasnica}@stuba.sk).*

Abstract: This paper introduces OPTIPLAN - a Matlab-based toolbox for formulating, solving, and simulating model predictive controllers (MPC) with embedded obstacle avoidance functionality. The toolbox offers a simple, yet powerful user interface that allows control researchers and practitioners to set up even complex control problems with just a few lines of code. OPTIPLAN fully automates tedious mathematical and technical details and let the user concentrate on the problem formulation. It features a rich set of tools to perform closed-loop simulations with MPC controllers and to visualize the results in an appealing way. From a theoretical point of view, OPTIPLAN tackles non-convex obstacle avoidance constraints in two ways: either by using binary variables or by resorting to suboptimal, but convex, time-varying constraints.

Keywords: collision avoidance, autonomous vehicles, model predictive control

1. INTRODUCTION

Collision-free control of autonomous vehicles is of a big interest nowadays and plays a major role in ensuring safety of the vehicles and the surrounding environment alike. Many control strategies ensuring collision-free operation of autonomous vehicles have been proposed in the literature with model prediction control (MPC) being the predominant technique, see, e.g. Yoon et al. (2009) and references therein. The popularity of MPC is mainly due to the fact that it can naturally incorporate constraints (including obstacle avoidance constraints) directly into the decision making process. Moreover, since MPC employs predictions of the future behavior of the system and its environment, it is straightforward to include prediction of moving obstacles into the problem, see, e.g., (Carvalho et al., 2015). Finally, MPC is an optimization-based control strategy and the computed control inputs are thus optimal with respect to some performance measure. Due to these advantages MPC has found its way into applications involving steering of autonomous vehicles (Keviczky et al., 2006), autonomous braking (Falcone et al., 2007), improvement of passangers' comfort (Elbanhawi et al., 2016), adaptive cruise control (Corona and De Schutter, 2008), and control of racing cars (Liniger et al., 2015) to name just a few.

Although a plethora of MPC-based control algorithms for collision-free trajectory planning and following exists in the literature, the respective authors rarely provide their software implementation to general public (let alone under free/open-source terms). This leaves interested control researchers and/or practitioners with the difficult task of implementing theoretical algorithms on their own using general-purpose optimization modeling tools, such as YALMIP (Löfberg, 2004), CVX (Grant and Boyd, 2014), or ACADO (Houska et al., 2011). Going the manual way, however, opens doors to erroneous and/or suboptimal formulations.

The objective of OPTIPLAN is to provide a free, open-source tool for MPC-based control of autonomous vehicles with embedded obstacle avoidance capabilities. The toolbox features a simple, yet versatile user interface that allows even non-experts to set up MPC problems using just few lines of code. The toolbox then automatically takes care of tedious mathematical and technical details as to provide an efficient formulation of the underlying optimization problem. In particular, OPTIPLAN allows for two different formulations of obstacle avoidance constraints. The first one, discussed in Section 3.1, employs binary variables to avoid obstacles in an optimal fashion. The downside is that it yields a non-convex mixed-integer problem that can be difficult to solve. As an alternative, the toolbox allows to avoid obstacles in a suboptimal way by using time varying constraints, cf. Section 3.2. The advantage here is that the underlying constraints are convex and allow for a simpler optimization problem to be solved as every sampling instant.

Although all theoretical concepts employed in this paper are well known (see, e.g. Williams (1993) and Frasch et al. (2013)), we believe the presented tool is of interest both to the research community, as well as to control practitioners willing to use MPC in their applications. OPTIPLAN exposes a powerful functionality using a user-friendly interface and hides (and fully automates) technical tasks. This allows its users to concentrate on problem

10.1016/j.ifacol.2017.08.057

formulation rather than on cumbersome math and/or Matlab programming.

1.1 Notation

We denote by $\mathbb{R}^n$ and $\mathbb{R}^{n\times m}$ the set of real-valued n-dimensional vectors and $n \times m$ matrices, respectively. $\mathbb{N}_a^b$ denotes the set of consecutive integers from a to b, i.e., $\mathbb{N}_a^b = \{a, a+1, \dots, b\}$ for $a \leq b$. $\|z\|_Q^2 := z^\intercal Q z$ with $z \in \mathbb{R}^n$ and $Q \in \mathbb{R}^{n\times n}$, $Q \succeq 0$ is the weighted squared 2-norm of the vector z.

2. PROBLEM SETUP

OPTIPLAN allows to easily synthesize, solve, and simulate MPC-based feedback laws for autonomous vehicles (robots, quadcopters, UAVs, etc.) that are required to avoid obstacles. The dynamics of the controlled vehicle (which will be referred to as an *agent*), is governed by discrete-time state-update and output equations of the form

$$x_{k+1} = f(x_k, u_k),\ y_k = g(x_k, u_k), \tag{1}$$

where $x \in \mathbb{R}^{n_\mathrm{x}}$ is the agent's state vector, $u \in \mathbb{R}^{n_\mathrm{u}}$ is the vector of control inputs, and $y \in \mathbb{R}^{n_\mathrm{y}}$ is the vector of agent's outputs. Without loss of generality we will assume that the output vector corresponds with the agent's position in the n_y-dimensional Euclidian space.

The task is to devise a feedback controller that manipulates the control inputs u in such a way that:

(1) state, input, and output constraints of the form
$$x \in \mathcal{X},\ u \in \mathcal{U},\ y \in \mathcal{Y}, \tag{2}$$
are enforced;
(2) the agent avoids obstacles $\mathcal{O}_j \subset \mathbb{R}^{n_\mathrm{y}}$, i.e., that $y \notin \mathcal{O}_j$ $\forall j \in \mathbb{N}_1^{n_\mathrm{obs}}$;
(3) the agent tracks a user-specified trajectory y_ref as closely as possible.

OPTIPLAN allows for different types of the dynamics in (1) as long as the system is controllable. In particular, it supports linear time invariant dynamics with $f(x,u) := Ax + Bu$ and $g(x,u) := Cx + Du$, as well as generic nonlinear functions $f : \mathbb{R}^{n_\mathrm{x}} \times \mathbb{R}^{n_\mathrm{u}} \to \mathbb{R}^{n_\mathrm{x}}$ and $g : \mathbb{R}^{n_\mathrm{x}} \times \mathbb{R}^{n_\mathrm{u}} \to \mathbb{R}^{n_\mathrm{y}}$. Moreover, we assume that the constraint sets in (2) are polyhedra of corresponding dimension, and the obstacles $\mathcal{O}_j$ are all polytopes (i.e., bounded polyhedra) in the half-space representation:

$$\mathcal{O}_j = \{y \mid \alpha_{i,j}^\intercal u \leq \beta_{i,j},\ i = 1, \dots, m_j\}, \quad \forall j \in \mathbb{N}_1^{n_\mathrm{obs}}. \tag{3}$$

Here, m_j is the number of half-spaces that define the j-th obstacle.

Given the input data (the current value of the agent's state $x(t)$, the dynamics in (1), the constraints in (2), and the obstacles in (3)), the MPC problem that OPTIPLAN solves is given by

$$\min_{u_0,\dots,u_{N-1}} \sum_{k=0}^{N-1} \left(||y_k - y_{\mathrm{ref},k}||_{Q_\mathrm{y}}^2 + ||u_k - u_{\mathrm{ref},k}||_{Q_\mathrm{u}}^2 \right) \tag{4a}$$
$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k), \tag{4b}$$
$$y_k = g(x_k, u_k), \tag{4c}$$
$$x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ y_k \in \mathcal{Y}, \tag{4d}$$
$$y_k \notin \mathcal{O}_j,\ \forall j \in \mathbb{N}_1^{n_\mathrm{obs}}, \tag{4e}$$
$$x_0 = x(t), \tag{4f}$$

with (4b)−(4e) imposed for $k = 0, \dots, N-1$. In (4a), $y_{\mathrm{ref},k}$ and $u_{\mathrm{ref},k}$ are the (possibly time-varying) references for the agent's outputs and inputs to be followed, respectively. In case of no preview of future references being available, $y_{\mathrm{ref},k} = y_\mathrm{ref}$ and $u_{\mathrm{ref},k} = u_\mathrm{ref}$ $\forall k \in \mathbb{N}_0^{N-1}$ is assumed. Moreover, Q_y and Q_u are weighting matrices used to tune the performance. The optimization is performed with respect to $u_0, \dots, u_{N-1}$. Then, in the spirit of a receding horizon implementation, only the first optimized input, i.e., $u_0^\star$ is implemented to the system in (1) and the whole procedure is repeated at a subsequent time instant for a new value of the initial condition in (4f).

If the obstacle avoidance constraints (4e) are disregarded, the optimization problem (4) becomes a "standard" MPC problem that can be readily formulated with off-the-shelf tools such as YALMIP, CVX, or ACADO, and solved using a plethora of free or commercial solvers such as GUROBI, CPLEX, `quadprog`, `fmincon`, depending on the type of the dynamics in (4b) and (4c), cf. (1).

However, with the constraint (4e) in place, the task becomes more challenging because, even though the obstacles $\mathcal{O}_j$ are assumed to be convex, the constraint $y \notin \mathcal{O}_j$ is non-convex. Although a manual handling of such a non-convex constraint is possible, it is cumbersome, error-prone and, if not done in an optimal fashion, can negatively impact the complexity and thus the runtime of the optimization. The underlying objective of OPTIPLAN is therefore to automate the task of formulating and solving non-convex MPC problems as in (4) with a minimal intervention from the user. The mathematical fundaments of two different ways of formulating the obstacle avoidance constraints (4e) are presented in the next section.

3. TACKLING OBSTACLE AVOIDANCE CONSTRAINTS

In this section, we review to methods of formulating obstacle avoidance constraints in (4e). The first method is based on using binary variables and thus results in (4) being a mixed-integer optimization problem. Although such problems are non-convex and NP-hard, efficient off-the-shelf solvers can tackle them for at least a modest number of obstacles. On the other hand, the mixed-integer formulation provides an optimal way of avoiding obstacles.

The second method avoids binary variables by heuristically choosing the direction from which to avoid the obstacle (either from the left or from the right), followed by employing time-varying output constraints. Although this leads to just a suboptimal trajectory, the advantage is that the constraints remain convex.