**Noname manuscript No.**
(will be inserted by the editor)

# Model predictive cooperative localization control of multiple UAVs using potential function sensor constraints*

## A workflow to create sensor constraint based potential functions for the control of cooperative localization scenarios with mobile robots

**Jan Dentler* · Somsundar Kannan · Souad Bezzaoucha · Miguel A. Olivares-Mendez · Holger Voos**

Received: date / Accepted: date

**Abstract** The global localization of multiple mobile robots can be achieved cost efficiently by localizing one robot globally and the others in relation to it using local sensor data. However, the drawback of this cooperative localization is the requirement of continuous sensor information.

Due to a limited sensor perception space, the tracking task to continuously maintain this sensor information is challenging. To address this problem, this contribution is presenting a Model Predictive Control ($MPC$) approach for such cooperative localization scenarios. In particular, the present work shows a novel workflow to describe sensor limitations with the help of potential functions. In addition, a compact motion model for multi-rotor drones is introduced to achieve $MPC$ real-time capability. The effectiveness of the presented approach is demonstrated in a numerical simulation, an experimental indoor scenario with two quadrotors as well as multiple indoor scenarios of a quadrotor obstacle evasion maneuver.

**Keywords** Localization and navigation in multi-robot systems · Distributed robotic systems operating on land, sea and air · Multi-robot and multi-vehicle motion coordination · Model predictive control · Sensor constrained control · Unmanned aerial vehicle · Quadrotor

*This work was supported by FNR "Fonds national de la Recherche" (Luxembourg) through AFR "Aides à la Formation-Recherche" Ph.D. grant scheme No. 9312118.

Jan Dentler (corresponding author)
Tel.: +123-45-678910
E-mail: jan.dentler@uni.lu

Somasundar Kannan · Souad Bezzaoucha · Miguel A. Olivares-Mendez · Holger Voos
{somasundar.kannan, souad.bezzaoucha, miguel.olivaresmendez, holger.voos}@uni.lu
SnT - Interdisciplinary Centre for Security, Reliability and Trust
University of Luxembourg
6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg

## 1 Introduction

During the last years, there has been a dramatic increase in the use of Unmanned Aerial Vehicles ($UAV$s) for all kind of applications such as surveillance, aerial photography, transport, etc. However, the fast dynamics of these systems and the extended operational space makes their autonomous piloting a challenging task. The recent development [1] of applications for such systems targets not only autonomous flying of single $UAV$s, but also the coordinated interaction of multiple $UAV$s or of $UAV$s and ground robots. One essential task herein is the precise localization of these $UAV$s and robots in their environment.

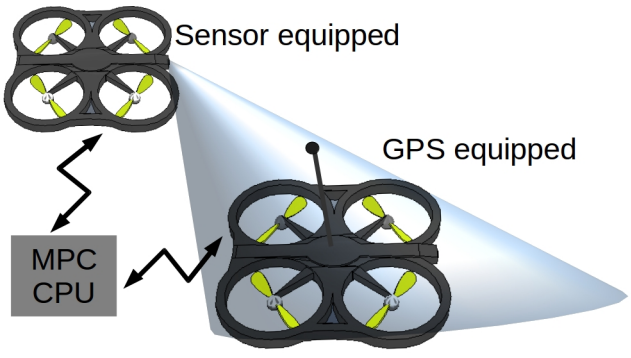


**Fig. 1** Cooperative localization scenario

The precise global navigation of $UAV$s though typically requires expensive specialized equipment such as differential $GPS$ in outdoor applications or laser-/RF-based global positioning systems in indoor applications. For cooperative mobile robot scenarios, one idea is therefore to deploy a re-

duced number of mobile robots which are equipped with the costly global localization system. The broadcasting of their global position allows all other $UAV$s to determine their own global position based on the relative position to these $UAV$s. The determination of the relative position can be achieved by less expensive onboard sensors such as optical sensors as shown in Fig. 1 in a $UAV$ scenario. The major problem of this approach is the necessity to continuously detect and track the globally localized robots with sufficient accuracy. This is further exacerbated by very dynamical $UAV$ scenarios and the onboard sensor limitations. Accordingly, the $UAV$ motion has to be controlled in order to keep the globally localized $UAV$s within the perception space of the onboard sensors. The major focus of this work is therefore to provide a central control strategy for such cooperative localization scenarios. To limit the scope of this paper, the estimation and localization itself are not addressed and a stable communication channel is assumed. Yet, it should be mentioned that all robot systems considered here do have internal controllers. A communication failure would thus only lead to missing localization data, but not to a complete system failure.

One method to handle such complex control scenarios is model predictive control. $MPC$ allows defining the control objective by means of an optimization problem. This so-called optimal control problem ($OCP$) is minimizing a given objective function subject to constraints. In general, the computational burden to solve an $OCP$ is high. Hence, the efficiency of the applied solver is limiting the complexity of the controlled real-time scenarios. Nevertheless, a central $MPC$ is well suited to control a small amount of robots. Such a central control simplifies the implementation of safety features and allows the computation of a global $OCP$ solution without considering additional consensus techniques. However, the presented methods here are also applicable for distributed controllers which will be addressed in future work.

One essential factor for fast $MPC$ is a compact description of the system's behavior. To tackle this problem, the first contribution of this work is a novel compact motion model for multi-rotor systems which is described in section 4. This simplistic motion model is based on a semi-nonlinear model for multi-rotor $UAV$s from previous work [2]. Due to its angle discontinuity problem, as shown experimentally in section 4.1, this previous model is not suitable for sensor tracking as considered within this work. To address this issue, section 4.2 is presenting a model, where the typical orientation description with a single yaw $\psi$ angle is replaced by a direction vector description. For validation, section 4.3 is showing the $MPC$ of a real $AR.Drone\,2.0$ quadrotor based on the derived direction vector model. The same modeling approach can also be adapted to other velocity controlled mobile robot systems.

Another difficulty of $MPC$ is the translation of the considered control scenario into an $OCP$. For this purpose, the second major contribution of this paper is a workflow that allows to represent sensor limitations in the form of potential functions, as presented in section 5. The considered use case is a sensor tracking scenario which is introduced in subsection 5.1. In this scenario, a quadrotor is controlled to keep an object within the cone-shaped perception space of the attached sensor. In this context, the sensor limitations are described as inequality constraints. These are subsequently transformed into weakened constraints that just appear in the $OCP$ cost function, as shown in section 5.2. This transformation is executed with unit steps and is recommended to maintain a low complexity which facilitates the mathematical and graphical validation of the resulting potential function. To improve the properties of the potential function for gradient and Newton based $OCP$-solvers, section 5.3 is dedicated to the introduction of artificial gradients in undesired regions of the potential function. As last step, section 5.4 is showing how the potential function is finally transformed into an analytical function by approximating all unit step functions by sigmoids.

For the experimental validation in the laboratory, additional safety constraints are introduced in section 6, based on the described workflow. This includes further potential functions to avoid collision in section 6.1 and to limit the operational space in section 6.2. Section 7 is finally presenting the validation of the deduced potential functions in the use case scenario and the chosen control parameters. The numerical validation is described in section 7.1, using the simulation environment $V\text{-}REP$. Section 7.2 is subsequently showing the results of the proposed control approach on real $AR.Drone\,2.0$ quadrotors. In the following section 7.3, the influence of obstacles, utilized constraints and different initial conditions is discussed on the basis of a set of collision avoidance scenarios in which a $UAV$ is evading an obstacle while tracking a target.

Finally, conclusion and future perspective are given in section 8.

## 2 Related Work

In order to solve a cooperative sensor based robot localization, the problem can be divided into a self-localization and a position tracking problem. The present work is focusing on providing a control strategy for the position tracking problem.

To be able to localize a robot by another robot while executing tasks, both robots have to be controlled in a cooperative manner. There is extensive literature regarding such cooperative control scenarios. A comprehensive overview of the subject is given by [3] and [4]. [3] is discussing the theory of swarm mechanics and interaction constraints while

[4] is providing a survey on formation control and coordination of multiple robots. According to [4] the coordination and control algorithms can be classified in leader-follower, behavioral based, virtual structure, graph based and potential field based approaches. The leader-follower principle is a well-established approach for non-holonomic mobile robots [5], [6], particularly regarding decentralized controllers in order to maintain the flexibility of a distributed system. These decentralized controllers are typically based on feedback linearization [7], [8] or backstepping and can be adapted to different tasks by switching the control law [9]. The same control approaches can also be found in behavioral [10], virtual structure [11], [12], graph theory based [13], [14], [15] and artificial potential [16],[17] based control approaches.

A more generic tool for multi-robot control is $MPC$ which is based on formulating the control scenario as optimization problem. One typical example of $MPC$ for $UAV$s is trajectory tracking in formation flight while considering collision avoidance constraints. Examples of a centralized $MPC$ of cooperative control scenarios is given in [18] and [19]. In [18] a leader-follower mode is used to perform airplane formation flight with collision avoidance using nonlinear $MPC$. For this purpose [18] compares centralized, sequential decentralized and fully decentralized methods of nonlinear $MPC$. [19] is presenting a non-convex $MPC$ for cooperative control. Here, the first objective is to tackle collision avoidance while the secondary performance objective is to deal with the quality of the collision-free trajectory. Examples of the use of decentralized $MPC$ is given by [18], [20], [21], [22]. The considered scenario in [20] is formation control of a multi-vehicle system with collision avoidance and input constraints. For this purpose a feedback linearization controller is integrated with $MPC$. The application of [21] is trajectory tracking of aerial robots under formation constraints using decentralized $MPC$. In [22] a decentralized linear time-varying $MPC$ is used for formation control of multiple $UAV$s using a leader-follower approach.

The generality and the high control performance of $MPC$ come with a high computational burden. Hence, the real-time capability is a crucial aspect of $MPC$ which has led to a variety of fast optimization algorithms to minimize the related computational effort. A theoretically well-established and widely used fast $MPC$ algorithm is sequential quadratic programming ($SQP$) in combination with Newton-type solvers with e.g. Gauß-Newton or Broyden-Fletcher-Goldfarb-Shannon ($BFGS$) Hessian approximation. A comprehensive framework with a wide variety of related algorithms is $ACADO$ [23]. The computational efficiency and real-time feasibility for fast mobile robot systems have been validated experimentally, as for example in collision avoidance scenarios with an aerial manipulator [24] under use of $BFGS$. A computationally efficient non-$SQP$ variation of a Newton-type method is the continuation generalized minimal residual ($CGMRES$) method as presented in [25]. Its underlying concept is introduced in section 3. A compact version in C++ code is freely available under [26]. The low computational burden of $CGMRES$ makes it particularly suitable to control fast systems, such as e.g gasoline engines [27], hover crafts [28] and Eco cruise control scenarios [29]. To increase the numerical stability, the condensed multiple shooting derivative $CMSCGMRES$ has been developed in [30], [31]. In the previous publication [2], $CMSCGMRES$ has been successfully implemented to control a commercial quadrotor. The low computation time and real-time capability of $CMSCGMRES$ has been confirmed experimentally for the given scenario. For this reason, this contribution is also based on $CMSCGMRES$. To reduce the implementational effort of additional inter-robot communication and consensus mechanisms and to compute a globally optimal solution, this contribution is computing the $MPC$ centrally. Yet, in order to maintain the modularity of the distributed system in the central $MPC$ scheme, the modularization scheme from previous work [32] is utilized in the form of the $DENMPC$ framework, as published in [33] (more details are given in section 3).

Within this work the cooperative control for a localization scenario with limited sensor perception is discussed. In the context of formation control the problem of a limited sensor perception space is critical along with collision avoidance and trajectory tracking. One approach for vision sensors is visual servoing based on optical flow or features, as shown in [34], [35] and [36]. For traditional backstepping controllers, sensor perception limits are addressed by switching the robot's formation control according to the compliance with the sensor constraints [37]. Another approach is to compute the optimal boundary trajectories to satisfy the sensor constraint and track these, as shown for non-holonomic robots in [38] or for visual servoing in [39], [40]. Nevertheless, the task dependency of the control laws makes it challenging to formulate control laws for complex scenarios with constraints. One way to avoid this loss of generality is to use $MPC$ which allows defining tasks and constraints as optimization problem in a generic way. An example for a cooperative $MPC$ using barrier function constraint handling with sensor and collision avoidance constraint is given in [41] . In [42] an aerial manipulator is presented with a camera attached to the end-effector. The camera is controlled using a stochastic $MPC$ method for visual servoing in order to keep the target in the field of view. [43] is presenting the direct implementation of sensor constraints in $MPC$ for holonomic mobile manipulators. The handling of constraints in $MPC$ itself is a wide field of research. The major difficulty in $MPC$ is how constraints are handled within the $MPC$ solver. An overview and benchmark of computationally efficient inequality constraint handling techniques with $CGMRES$ is given in [44]. The disadvantage of these sim-

plistic constraint handling techniques, as for example auxiliary variable and logarithmic barrier method, is that a violation leads to an infeasible $OCP$ and accordingly to a crash of the $MPC$ solver. This is particularly problematic for fast $MPC$ solvers, as they do not consider invalid values (*inf*, *nan*) within the prediction horizon and do therefore not automatically recover from an infeasible state. If a small constraint violation can be accepted, one way to avoid this problem is the use of weakened constraints. A weakened constraint is approximating the inequality constraint switching behavior by an analytical function (e.g. $sigmoid$ [45], $tanh$) which leads to a potential function. A comprehensive study on such weakened constraints in combination with $MPC$ for multi $UAV$ control strategies is given in [46]. The provided examples are collision avoidance, area exploration and formation flying. The same approach has been successfully implemented for collision avoidance of a $MPC$ controlled quadrotor system in previous work [2].

To conclude, in order to solve the position tracking problem of cooperative localization scenarios, this work is presenting a workflow to formulate sensor constraints as potential functions. To be more flexible than feedback linearization and backstepping control approaches in task and constraint description, a $MPC$ approach is applied. In order to achieve real-time capability the $MPC$ problem is solved by a central $CMSCGMRES$ approach. To maintain the modularity of the MPC with respect to the distributed problem, this approach is combined with the $MPC$ modularization technique, presented in [32]. The prediction model for the $MPC$ is based on the quadrotor model from the previous contribution [2]. An introduction of the utilized $MPC$ approach is given in the following section.

## 3 Model Predictive Control Principle and Modularization

The idea of $MPC$ is to compute optimal controls for a given system objective within a receding horizon as shown in Fig. 2. For this purpose, the current system state is measured at each control update time instance $(t_k, t_{k+1}, ...)$. From each of these measured states, the future systems behavior can be predicted by means of a system model within a given horizon ($0 \leq \tau \leq T$). The controls for this horizon are then determined to minimize the error of the future system behavior to a given target behavior. After determining the optimal controls, $\mathbf{u}(t_k) = \mathbf{u}(\tau_0)$ is applied to the system and the horizon is shifted by $\Delta t$. The control loop is closed by a new measurement and prediction at time instance $t_{k+1}$ ($\Delta t$ later). The computation of the optimal controls within the prediction horizon can be formulated as optimization problem, a so called optimal control problem. In this work, $OCP$s of

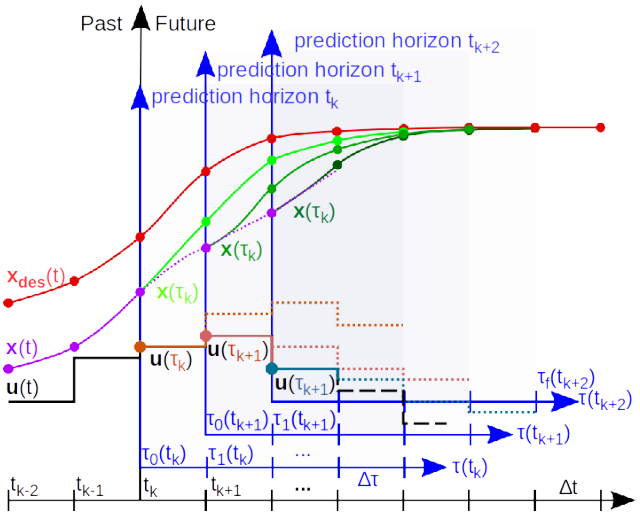


**Fig. 2** MPC operation cycle

the form

$$\min_{\mathbf{u}} \; J = \int_{\tau_0}^{\tau_f} l(\mathbf{x}, \mathbf{u}, \tau)\, \mathrm{d}\tau \tag{1}$$
$$\text{s.t.} \quad \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \tau)$$
$$\mathbf{x}_0 = \mathbf{x}(\tau_0)$$

are considered. The desired behavior is defined by the cost function $J$ which is consisting of an integral cost term. It is subjected to the prediction model dynamics $\mathbf{f}(\mathbf{x}, \mathbf{u}, \tau)$. The initial state condition $\mathbf{x}_0$ is representative for the measurement at each control update interval. A common way to solve $OCP$s in real-time applications is to derive the Hamiltonian for (1)

$$H(\mathbf{x}, \mathbf{u}, \tau) = l(\mathbf{x}, \mathbf{u}, \tau) + \boldsymbol{\lambda}^\top \mathbf{f}(\mathbf{x}, \mathbf{u}, \tau), \tag{2}$$

with the Lagrange multipliers $\boldsymbol{\lambda}$. $\boldsymbol{\lambda}$ can then be computed via backward integration from $\mathbf{0}$ over the horizon using

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial H(\mathbf{x}, \mathbf{u}, \tau)}{\mathbf{x}}. \tag{3}$$

The controls are then typically determined by solving the first order optimality condition

$$0 = \frac{\partial H(\mathbf{x}, \mathbf{u}, \tau)}{\partial \mathbf{u}} = \frac{\partial l(\mathbf{x}, \mathbf{u}, \tau)}{\partial \mathbf{u}} + \boldsymbol{\lambda}^\top \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u}, \tau)}{\partial \mathbf{u}}. \tag{4}$$

There is a wide variety of methods to solve (4), whereas for real-time $MPC$ common approaches are gradient (e.g. $GRAMPC$ [47]) or Newton-type (e.g. Gauß-Newton in $ACADO$ [23]) line search methods.

Within this work, a continuation generalized minimal residual method $CGMRES$ derivative is used which has been already successfully applied in previous work [32], [2]. The continuation idea is to look at the continuous closed loop dynamics of the stabilized system

$$\dot{H}(\mathbf{x}, \mathbf{u}, \tau) = \xi_u H(\mathbf{x}, \mathbf{u}, \tau) \tag{5}$$

which can then be used to approximate $\dot{\mathbf{u}}$

$$\dot{\mathbf{u}} = H_{\mathbf{u}}^{-1} \left( \xi_u H - H_{\mathbf{x}} \dot{\mathbf{x}} - H_t \right). \tag{6}$$

$\dot{\mathbf{u}}$ can be computed efficiently with a $GMRES$ method under use of a forward difference approximation of $H_u^{-1}$. The controls $\mathbf{u}$ are finally gained by integrating $\mathbf{u}$ from the previous time step with $\dot{\mathbf{u}}$

$$\mathbf{u}\left[t_k + 1\right] = \mathbf{u}\left[t_k\right] + \dot{\mathbf{u}}\left[t_k\right] \Delta t. \tag{7}$$

More detailed information about the $CGMRES$ algorithm can be found in [25]. By making the continuation assumption (5) also for the states $\mathbf{x}$, a multiple shooting method can be derived to increase numerical stability. This increases the problem dimension, as the corresponding equation to (5) for the inputs $\mathbf{u}$ has also to be solved for the predicted system states within the horizon. An additional condensing addresses this issue by reducing the problem dimension again. A detailed overview of the $CMSCGMRES$ method is given in [31]. The major advantage of using the continuation approach is the low computation time of the $CGMRES$ method which makes it particularly interesting for real-time $MPC$, as already discussed in previous work [2].

Within this work, a modularization of $MPC$ for cooperative control scenarios is applied, as presented in [32]. For cooperative scenarios with two entities ($agent0 : x_0, u_0$ and $agent1 : x_1, u_1$) a cost function can have the form

$$\begin{aligned} & l\left(\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_1, \mathbf{u}_1\right) \\ &= l_0\left(\mathbf{x}_0, \mathbf{u}_0\right) + l_1\left(\mathbf{x}_1, \mathbf{u}_1\right) + l_c\left(\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_1, \mathbf{u}_1\right), \end{aligned} \tag{8}$$

where $l_0$ and $l_1$ represent tracking cost functions, while $l_c$ is defining some interaction costs. Under this assumption the optimality condition 6 yields to

$$\begin{aligned} 0 = & \frac{\partial l_0\left(\mathbf{x}_0, \mathbf{u}_0, \tau\right)}{\partial \mathbf{u}_0} + \frac{\partial l_1\left(\mathbf{x}_1, \mathbf{u}_1, \tau\right)}{\partial \mathbf{u}_1} \\ & + \frac{\partial l_c\left(\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_1, \mathbf{u}_1\right)}{\partial \mathbf{u}_0} + \frac{\partial l_c\left(\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_1, \mathbf{u}_1\right)}{\partial \mathbf{u}_1} \\ & + \boldsymbol{\lambda}_0^{\top} \frac{\partial \mathbf{f}_0\left(\mathbf{x}_0, \mathbf{u}_0, \tau\right)}{\partial \mathbf{u}_0} + \boldsymbol{\lambda}_1^{\top} \frac{\partial \mathbf{f}_1\left(\mathbf{x}_1, \mathbf{u}_1, \tau\right)}{\partial \mathbf{u}_1} \end{aligned} \tag{9}$$

With the optimality condition (9) $l_c$ will influence $u_0$ through $\frac{\partial l_c(\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_1, \mathbf{u}_1)}{\partial \mathbf{u}_0}$ and $u_1$ through $\frac{\partial l_c(\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_1, \mathbf{u}_1)}{\partial \mathbf{u}_1}$. The modularization allows to switch off the influence of $l_c$ on e.g. $u_1$ by neglecting $\frac{\partial l_c(\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_1, \mathbf{u}_1)}{\partial \mathbf{u}_1}$ in the optimality condition (9). This is useful, if one central $MPC$ is used to control the complete cooperation scenario, but some interaction tasks shall just influence one drone (e.g. the use case scenario of this work). The further advantage of this centralized approach is its high performance, as not just the measured, but also the predicted future system states are taken into consideration. In the $OCP$ the exclusion of cost derivatives (e.g. with respect to $u_0$) will be marked by an index $\setminus\{u_0\}$, e.g.

$$l^{\setminus\{u_0\}}\left(\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_0, \mathbf{u}_0, \tau\right) \tag{10}$$

To finally apply a $MPC$ in a mobile robot scenario, a prediction model is required. For real-time applications the complexity of this prediction model is crucial for the performance of the $MPC$ solver. Hence, a very compact model is preferable which is developed in the next section.

## 4 Direction Vector Prediction Model and Validation

Most mobile robots are designed to move in a planar space. Their position is defined in a $xy$-plane and the height of this plane ($z$). Accordingly, their attitude is defined by the rotation angle $\Psi$ around the plane normal vector. This description does not only fit for most wheeled ground robots, but can also be applied to multi-rotor unmanned aerial vehicles. Multi-rotor $UAV$s are typically operated around their static equilibrium. By assuming that rotations around $x$ (roll) and $y$ (pitch) are so small to be be neglected, the quadrotor attitude can be linearized. This yields to a hover controller [48], as implemented in most commercial multi-rotor $UAV$s. As a result, the pose can be described by a single yaw angle $\Psi$, representing the rotation around the $z$ axis. To address the modeling of such systems, this section is introducing a modeling method for hover controlled multi-rotor systems. This model has furthermore the advantage to contain all elements from which omnidirectional and unidirectional ground robot motion models can also be derived.

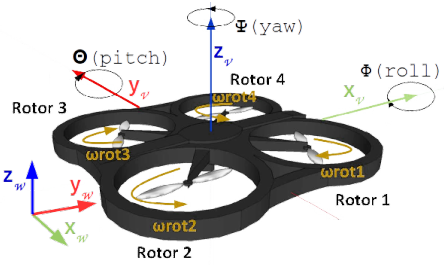


**Fig. 3** AR.Drone 2.0[1] coordinate frame definition

To derive the generalized multi-rotor model, the coordinate frames, given in Fig. 3 are considered. A standard approach is to model the system's behavior in the state-space and to give the system function as an Ordinary Differential Equation ($ODE$). The state vector $\mathbf{x}$ of such a system

$$\mathbf{x}\left(t\right) = \left[x_{\mathcal{G}}\left(t\right), y_{\mathcal{G}}\left(t\right), z_{\mathcal{G}}\left(t\right), \Psi\left(t\right), \dot{x}_{\mathcal{V}}\left(t\right), \dot{y}_{\mathcal{V}}\left(t\right)\right]^{\top} \tag{11}$$

is composing $x_{\mathcal{G}}$, $y_{\mathcal{G}}$, $z_{\mathcal{G}}$ position in the global frame $\mathcal{G}$, the $\Psi_{\mathcal{G}}$ rotation around $z_{\mathcal{G}}$ and the forward $\dot{x}_{\mathcal{V}}$ and sideward $\dot{y}_{\mathcal{V}}$ velocity in the vehicle frame $\mathcal{V}$. Under use of the forward, sideward, upward and heading velocity as input

$$\mathbf{u}\left(t\right) = \left[u_f\left(t\right), u_s\left(t\right), u_z\left(t\right), u_{\Psi}\left(t\right)\right]^{\top}, \tag{12}$$

the system dynamics $\mathbf{f}\left(\mathbf{x}\left(t\right),\mathbf{u}\left(t\right)\right)$ can be approximated by a semi-nonlinear state-space model [2]

$$\dot{\mathbf{x}}\left(t\right)=\mathbf{f}\left(\mathbf{x}\left(t\right),\mathbf{u}\left(t\right)\right) \tag{13}$$

$$\dot{\mathbf{x}}\left(t\right)=\begin{bmatrix}\dot{x}_{\mathcal{V}}\left(t\right)\cos\left(\Psi\left(t\right)\right)-\dot{y}_{\mathcal{V}}\sin\left(\Psi\left(t\right)\right)\\ \dot{x}_{\mathcal{V}}\left(t\right)\sin\left(\Psi\left(t\right)\right)+\dot{y}_{\mathcal{V}}\cos\left(\Psi\left(t\right)\right)\\ b_z\cdot u_z\left(t\right)\\ b_\Psi\cdot u_\Psi\left(t\right)\\ a_f\cdot\dot{x}_{\mathcal{V}}\left(t\right)+b_f\cdot u_f\left(t\right)\\ a_s\cdot\dot{y}_{\mathcal{V}}\left(t\right)-b_s\cdot u_s\left(t\right)\end{bmatrix}\begin{matrix}\left.\vphantom{\begin{matrix}a\\a\end{matrix}}\right\}\text{Map: }\mathcal{V}\rightarrow\mathcal{G}\\ \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\}\text{Linear model.}\end{matrix} \tag{14}$$

For low-cost quadrotors the velocity inputs are typically just related to an attitude displacement, e.g. increase pitch angle to increase forward velocity. Accordingly the inputs are not expressing the exact velocity in $\mathrm{m\,s^{-1}}$. The identification of the linear model parameters helps to describe this attitude control behavior and to accommodate such unnormed system inputs. Representatively for such low-cost mobile robotic systems, the quadrotor *AR.Drone* 2.0 is applied as example within this work.

### 4.1 Angle Discontinuity Effect on UAV Control

The attitude of mobile robots which are moving in a planar plane is typically defined by a single angle $\Psi$ on the interval

$$\Psi:=\{\Psi\in\mathbb{R}|-\pi<\Psi\leq\pi\}. \tag{15}$$

For a full rotation of the mobile robot, $\Psi$ is changing the sign between $\pm\pi$. By artificially limiting the angle on the interval (15) a discontinuity is introduced into the model of the angle. This is problematic, if the attitude is controlled by means of velocity or acceleration. To show the problematic behavior, a model predictive controller is applied to an *AR.Drone 2.0* quadrotor under use of the prediction model (11) with

$$\begin{aligned}&\left(b_z,b_\Psi,a_f,b_f,a_s,b_s\right)^\top\\&=\left(1,1.6,-0.5092,1.458,-0.5092,1.458\right)^\top\end{aligned} \tag{16}$$

The resulting trajectory is given in Fig. 4. To control the quadrotor attitude, traditionally the error $e_\Psi=\Psi_{des}-\Psi$ to the desired yaw angle $\Psi_{des}$ is minimized. The yaw angle plot in Fig. 4 shows a step in the desired angle from $\Psi_{des}\left(0\mathrm{s}\right)=\frac{\pi}{2}$ to $\Psi_{des}\left(2.8\mathrm{s}\right)=\pi$. To minimize $e_\Psi$, the controller is applying a positive angular velocity to converge asymptotically towards $\Psi_{des}=\pi$. As real mobile robotic systems are exposed to disturbance, the robot is likely to overshoot $\Psi_{des}=\pi$ which leads to a change of sign to $\Psi=-\pi$. This can be seen in Fig. 4 at $t\approx 5.4\mathrm{s}$. Hence, the quadrotor will again try to reach the desired value from the new angle $\Psi_{des}=-\pi$. As a result, positions close to $\Psi_{des}=\pm\pi$ cannot be stabilized which leads to a repetitive rotational movement.

The main use of angle $\Psi$ is the mathematical description for the mapping of the vehicle coordinate frame $\mathcal{V}$ to the

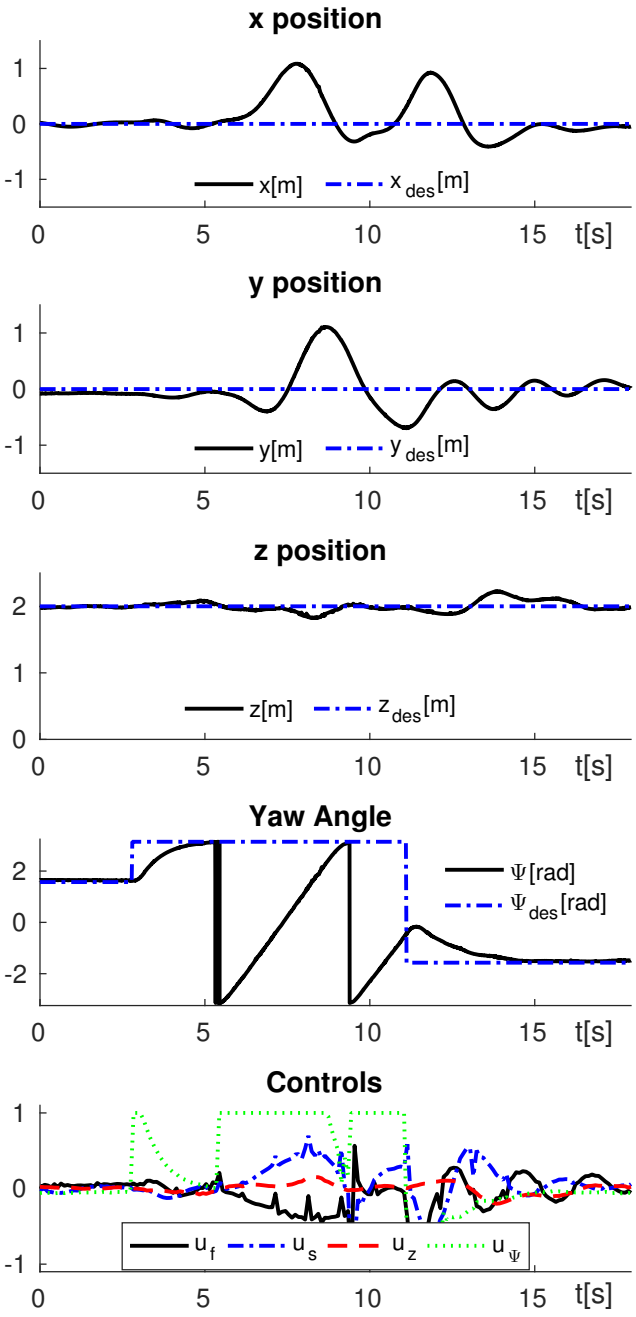


**Fig. 4** Angle discontinuity problem regarding the control trajectories of a real *AR.Drone* 2.0 quadrotor

global coordinate frame $\mathcal{G}$. This is typically done by means of a rotation matrix, as shown in (14). For the planar case the rotation matrix yields

$$\mathbf{p}_{\mathcal{V}}=\begin{bmatrix}\cos(\Psi)&-\sin(\Psi)\\ \sin(\Psi)&\cos(\Psi)\end{bmatrix}\mathbf{p}_{\mathcal{G}}=\mathbf{T}_{\mathcal{G}}^{\mathcal{V}}\mathbf{p}_{\mathcal{G}}\text{ with }\mathbf{p}=\begin{bmatrix}x&y\end{bmatrix}^\top. \tag{17}$$

To overcome the described angle discontinuity in the transformation, a continuous way to describe the robot attitude has to be used. One way to do so is to use a direction vector. For 3D-space this is well established in the form of quaternions $\mathbf{q}\in\mathcal{R}^4$ which consist of a real part $q_0$ and three imaginary parts $q_1$, $q_2$, $q_3$ to describe rotations

$\mathbf{q} =< q_0, q_1, q_2, q_3 >$. In state space models these four elements are treated as separate states. For real-time $MPC$ the related increase in the $OCP$ dimension is significant. For the considered robots, the pose control is only based on a rotation in the $2D$ $xy$-plane. Accordingly, a direction vector approach is sufficient to describe the attitude, as discussed in the next section 4.2.

## 4.2 Direction Vector Approach

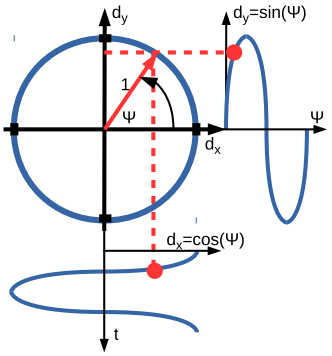


**Fig. 5** Unit circle

As previously discussed, the direction of a vector can be used to describe a robot attitude. Each position vector of the points on the unit circle can accordingly be used which is directly related to complex number theory. These vectors are uniquely defined by their projections onto the coordinate axis $x$ and $y$ as shown in Fig. 5. In comparison to a single angle description with yaw $\Psi$, this transformation is bijective. In the context of this work, the combination of these two projections ($d_x$ and $d_x$) is called direction vector $\mathbf{d}$. The direction vector $\mathbf{d}$ can be written in vector form

$$\mathbf{d} = \begin{bmatrix} d_x \\ d_y \end{bmatrix} = \begin{bmatrix} \cos(\Psi) \\ \sin(\Psi) \end{bmatrix} \quad \begin{matrix} d_x := \{d_x \in \mathbb{R} | -1 < d_x \leq 1\} \\ d_y := \{d_y \in \mathbb{R} | -1 < d_y \leq 1\} \end{matrix}. \tag{18}$$

As the direction vector is expressing the projection from the unit circle, $d_x$ and $d_y$ are fulfilling the circle constraint

$$1 = d_x^2 + d_y^2 \tag{19}$$

To receive the state space description, $\dot{\mathbf{d}}$ yields under use of (18) to

$$\dot{\mathbf{d}} = \begin{bmatrix} \dot{d}_x \\ \dot{d}_y \end{bmatrix} = \begin{bmatrix} -\sin(\Psi) \\ \cos(\Psi) \end{bmatrix} \dot{\Psi} = \begin{bmatrix} -d_y \\ d_x \end{bmatrix} \dot{\Psi} \tag{20}$$

This is intuitive, as the derivative $\dot{\mathbf{d}}$ has to be orthogonal to $\mathbf{d}$ to force $\mathbf{d}$ to stay on the unit circle.

The system input $u_\Psi$ is expressing the angular velocity $\dot{\Psi}$ in (20) with the constant factor $b_\psi$. Hence, the derivative yields to

$$\dot{\mathbf{d}} = \begin{bmatrix} -d_y \\ d_x \end{bmatrix} b_\Psi \cdot u_\Psi (t) \tag{21}$$

Transformation matrix (17) is transformed with (18) to

$$\mathbf{T}_\mathcal{G}^\mathcal{V} = \begin{bmatrix} d_x & -d_y \\ d_y & d_x \end{bmatrix} \tag{22}$$

To transform the system dynamics (14) into a direction vector model, $\Psi$ is substituted by the direction vector in the state vector

$$\mathbf{x}(t) = \left[x_\mathcal{G}(t), y_\mathcal{G}(t), z_\mathcal{G}(t), d_x(t), d_y(t), \dot{x}_\mathcal{V}(t), \dot{y}_\mathcal{V}(t)\right]^\top \tag{23}$$

Using the direction vector (18), the derivative (21) with $u_\Psi$ and the coordinate transformation (22), the system dynamics (14) finally leads to

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) = \begin{bmatrix} \dot{x}_\mathcal{V}(t) d_x(t) - \dot{y}_\mathcal{V}(t) d_y(t) \\ \dot{x}_\mathcal{V}(t) d_y(t) + \dot{y}_\mathcal{V}(t) d_x(t) \\ b_z \cdot u_z(t) \\ -d_y(t) \cdot b_\Psi \cdot u_\Psi(t) \\ d_x(t) \cdot b_\Psi \cdot u_\Psi(t) \\ a_f \cdot \dot{x}_\mathcal{V}(t) + b_f \cdot u_f(t) \\ a_s \cdot \dot{y}_\mathcal{V}(t) - b_s \cdot u_s(t) \end{bmatrix} \tag{24}$$

To track attitude $\mathbf{d}$, a simple quadratic penalty can be used

$$J_\mathbf{d} = (\mathbf{d}_{des} - \mathbf{d})^\top \begin{bmatrix} q_{dx} & 0 \\ 0 & q_{dy} \end{bmatrix} (\mathbf{d}_{des} - \mathbf{d}) \tag{25}$$

under the assumption, that numerical errors of the $MPC$ solver avoid the singular problem of opposing attitudes, e.g. $\mathbf{d}_{des} = [1, 0]^\top$ with $\mathbf{d} = [-1, 0]^\top$, or $\mathbf{d}_{des} = [0, 1]^\top$ with $\mathbf{d} = [0, -1]^\top$.

## 4.3 Experimental Validation of Direction Vector Approach

To validate the direction vector quadrotor model (24), the desired drone attitude is rotated anti-clock-wise in steps of $\Psi = \frac{\pi}{2}$. The $\Psi$ plot in Fig. 6 shows the desired and actual attitude of the system. In contrast to the previous instability at $\Psi = \pm\pi$ (Fig. 4), Fig. 6 is showing the desired asymptotic approaching of the desired trajectory. The oscillations in $x$, $y$, $z$ around the desired point are caused by disturbance. This includes airflow disturbance, modeling errors, numerical errors and the trade-off between energy optimality and position tracking. Hence, the direction vector approach is resolving the angle discontinuity problem stated in section 4.1. The proposed approach is a trade-off between the continuous attitude description, the computational effort, regarding the quaternion approach (4 states) and the standard angle description (1 state). Regarding the generality of the direction vector model, the same approach can be easily applied also to other planar robots with single angle attitude description. For example ground robots can be modeled by neglecting the $z$ component of (24) with $u_z = 0$. For unidirectional robots, the sideward movement can be neglected by setting

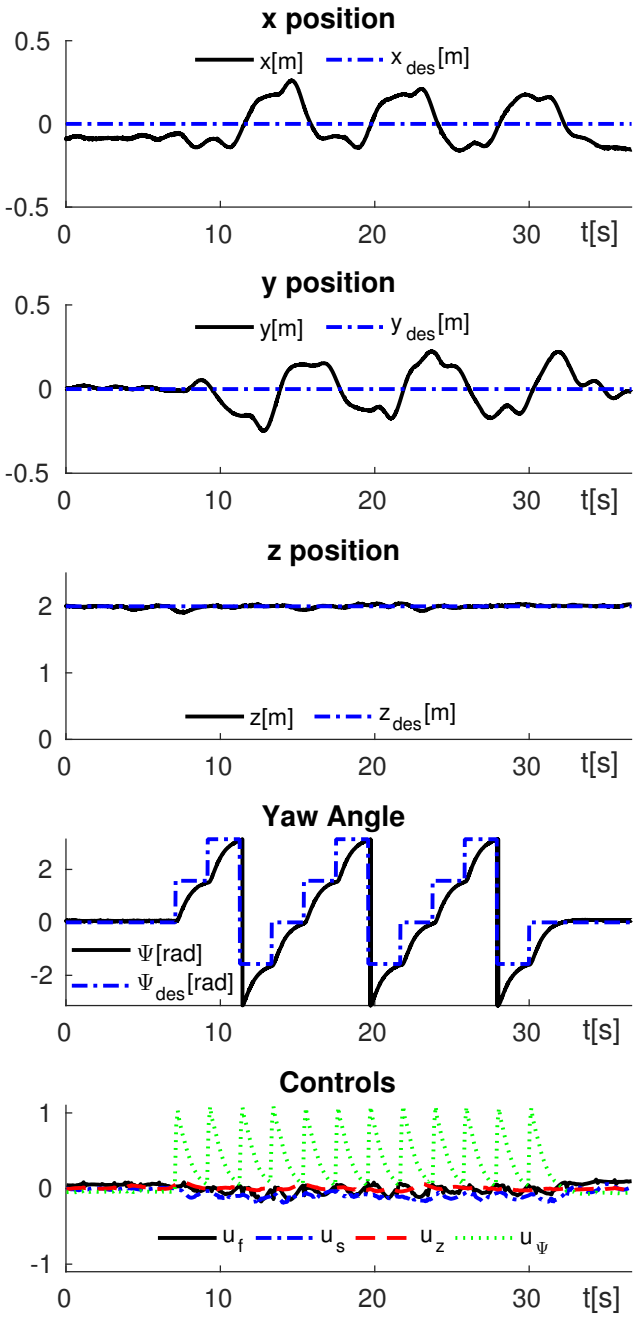


**Fig. 6** Experimental Validation of direction vector quadrotor model

$u_s = 0$. Based on the resulting direction vector based system dynamics, the next sections describe the application of the model in a sensor constrained model predictive control scenario.

## 5 Sensor Based Control With Potential Functions

One major difficulty to control complex tasks autonomously is the mathematical formulation of such tasks. A generic way to do so are inequality constraints. $MPC$ can take such constraints into consideration. In this work, they are considered in the $MPC$ as weakened constraints. This refers to the substitution of hard inequality constraints by a cost function, that imposes a repulsive behavior from a violation of the constraint. This implementation in the cost function equals to a potential function. In this section, a generic procedure to create such potential functions is presented. The formulation of a sensor constraint serves as example. In the example scenario an object is tracked with a sensor attached to a quadrotor. The sensor perception space is thereby shaped like a cone (e.g. ultrasonic distance sensor). How to describe this sensor limitation is shown in the following section.

### 5.1 Sensor Constraints

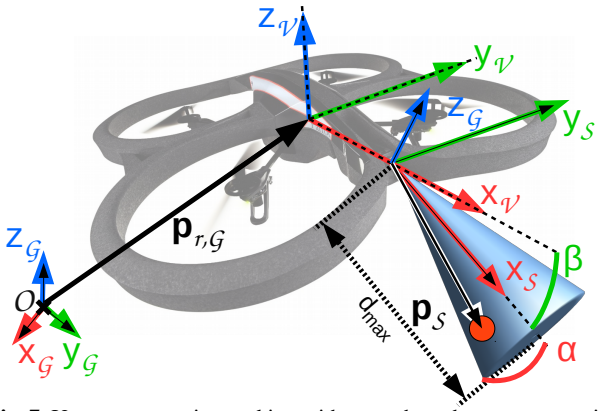


**Fig. 7** Use case scenario: tracking with cone shaped sensor perception of a quadrotor

The considered use case is an $AR.Drone\,2.0$ quadrotor with the presented dynamics (24). With the example of a cone-formed sensor perception space, the robot/sensor system can be illustrated as shown in Fig. 7. The first step to implement the sensor based control is, to formulate the perception space limitation in terms of constraints. For this purpose, a target object is defined in the sensor frame with the position vector $\mathbf{p}_{\mathcal{S}} \in \mathcal{S}$

$$\mathbf{p}_{\mathcal{S}} = \begin{bmatrix} x_{p\mathcal{S}} & y_{p\mathcal{S}} & z_{p\mathcal{S}} \end{bmatrix}^{\top} \in \mathcal{R}^3. \tag{26}$$

Regarding the position of the trackable object in the sensor frame $\mathbf{p}_{\mathcal{S}}$, the field of view of the sensor can be expressed by the constraints

$$0 \geq c_1\left(\mathbf{p}_{\mathcal{S}}, \alpha\right) = y_{p\mathcal{S}}^2 + z_{p\mathcal{S}}^2 - \left(x_{p\mathcal{S}} \sin(\alpha)\right)^2 \tag{27}$$

$$0 \geq c_2\left(\mathbf{p}_{\mathcal{S}}\right) = -x_{p\mathcal{S}}, \tag{28}$$

where (27) is representing a double cone. To receive a single cone perception space, constraint (28) is limiting the cone to the positive half plane of $x_{p\mathcal{S}}$. In contrast to just pointing the quadrotor into the target's direction, the formulation of the perception area offers more flexibility to the $MPC$ to optimize the energy consumption and to adapt the scenario to other constraints e.g. obstacles. The question of how to consider the sensor constraint (27)-(28) in an optimal control problem is described in the next section.

### 5.2 Potential Function Constraint Handling

Considering the constraints (27)-(28) as hard constraints is problematic, because their violation would lead to an infeasible optimization problem. In the sensor based tracking scenario this would be the case, if the object is outside the sensor perception space. Such a violation is possible due to disturbance or an infeasible initial pose. For this reason, the sensor constraints are designed as weakened constraints by imposing a repulsive behavior from constraint violation. One way to accomplished this is to add an additional penalty term to the $OCP$'s integral costs $l$. Due to the fact that $OCP$s are typically defined as minimization problems with the optimum $l = 0$, a constraint violation has to be penalized with a higher cost. An intuitive approach to translate the constraint $c \leq 0$ to the weakened constraint $l_c$ is therefore to penalize the compliant area with $l = 0$ and the constraint violation area with $l = 1$. This can be described by using a unit step $\epsilon$, such that

$$c \leq 0 \quad \Rightarrow \quad l_c = \epsilon\left(c\right). \tag{29}$$

For the cone constraint (27-28) this leads to

$$\begin{aligned} &l_c\left(\mathbf{p}_{\mathcal{S}}, \alpha\right) = \epsilon\left(c_2\left(\mathbf{p}_{\mathcal{S}}\right)\right) + \epsilon\left(c_1\left(\mathbf{p}_{\mathcal{S}}, \alpha\right)\right)\epsilon\left(-c_2\left(\mathbf{p}_{\mathcal{S}}\right)\right) \\ &= \epsilon\left(-x_{p\mathcal{S}}\right) + \epsilon\left(y_{p\mathcal{S}}^2 + z_{p\mathcal{S}}^2 - \left(x_{p\mathcal{S}}\sin(\alpha)\right)^2\right)\epsilon\left(x_{p\mathcal{S}}\right) \end{aligned} \tag{30}$$

$c_2$ is used to distinguish between the negative and positive half-space of $x_{p\mathcal{S}}$:

$$l_c(c_2\left(\mathbf{p}_{\mathcal{S}}\right) > 0 \rightarrow x_{p\mathcal{S}} \in \mathbb{R}^-) = 1 \tag{31}$$

$$l_c(c_2\left(\mathbf{p}_{\mathcal{S}}\right) \leq 0 \rightarrow x_{p\mathcal{S}} \in \mathbb{R}^+) = c_1\left(\mathbf{p}_{\mathcal{S}}, \alpha\right) \tag{32}$$

To initially use the unit step approximation (29) for the constraint transformation has proven to be particularly helpful regarding potential functions developed from nested constraints. More complex approximation functions lead to larger mathematical expression and reduce the readbility of the behavior of the potential function. Furthermore, the behavior of the developed potential function can be easily validated visually by plotting the areas with potential value $l_c = 0$ and $l_c = 1$. For the example of the weakened cone constraint (30), this results in the characterizing cone area with potential value $l_c = 0$ and the corresponding incompliant area outside with $l_c = 1$ as shown in Fig. 8.

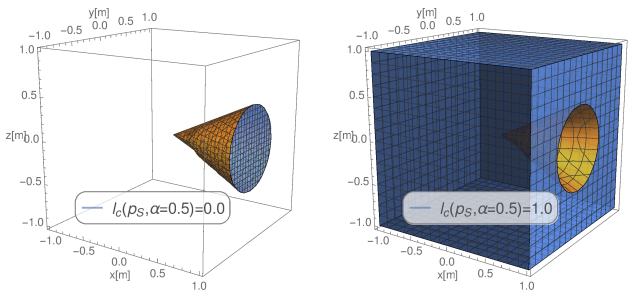


**Fig. 8** Unit step penalty function values of cone constraint)

### 5.3 Addressing Vanishing Gradient

The resulting cost function for the $xy$-plane is shown in the left plot of Fig. (9). As expected, the Fig. shows that the gradient satisfies

$$\nabla l_c\left(\mathbf{p}_{\mathcal{S}}, \alpha\right) = \mathbf{0} \qquad \forall c_1\left(\mathbf{p}_{\mathcal{S}}, \alpha\right) \neq 0 \tag{33}$$

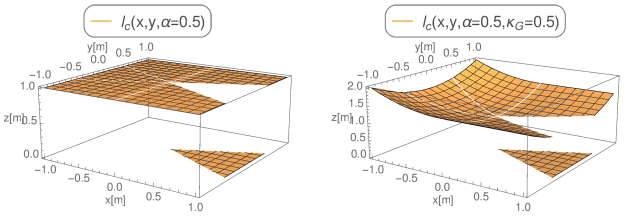


**Fig. 9** Unit step penalty function of cone constraint in the $xy$-plane)

As most approaches for searching the minimum of the defined cost function $l_c$ are based on a gradient descent, this property becomes problematic, as the convergence of the search algorithm cannot be guaranteed. To address this issue a cost slope can be added around the cone as shown in the right illustration of Fig. 9. To impose a gradient, a simple quadratic penalty can be added in the undesired regions. Hence, (30) can be extended to

$$\begin{aligned} &l_c\left(\mathbf{p}_{\mathcal{S}}, \alpha, \kappa_G\right) \\ &= \epsilon\left(-x_{p\mathcal{S}}\right)\left(1 + \kappa_G\left(x_{p\mathcal{S}}^2 + y_{p\mathcal{S}}^2 + z_{p\mathcal{S}}^2\right)\right) \\ &+ \epsilon\left(y_{p\mathcal{S}}^2 + z_{p\mathcal{S}}^2 - \left(x_{p\mathcal{S}}\sin(\alpha)\right)^2\right) \\ &\cdot\ \epsilon\left(x_{p\mathcal{S}}\right)\left(1 + \kappa_G\left(y_{p\mathcal{S}}^2 + z_{p\mathcal{S}}^2\right)\right), \end{aligned} \tag{34}$$

whereby parameter $\kappa_G$ is controlling the steepness of the gradient derivative. As a remark, gradient-based solvers typically require convex $OCP$ problems to ensure convergence. Naturally, most sensor perception spaces are convex. However this should be kept in mind for development of more complex constraints or the combination of multiple constraints.

### 5.4 Addressing Differentiability of the Potential Function

The second problematic property of the proposed unit step constraint translation (29) is that the derivative of the unit step is not an analytical function. Yet, most fast optimal control problem solvers require the differentiability of the cost function. To address this problem, the switching behavior of the unit step, with respect to the constraints, can be approximated by means of a sigmoid function. This has been already validated in previous work for a collision avoidance constraint [2] and is similar to the $\tanh$ approximation, as introduced in [46]. Accordingly, the unit step can be approximated with

$$\epsilon\left(c\right) \approx sig\left(c, \kappa_A\right) = \frac{1}{1 + e^{-\kappa_A c}}, \tag{35}$$

where $\kappa_A$ is a design parameter. The derivative of $sig\left(c,\kappa_A\right)$ can be determined analogously

$$\frac{\partial sig\left(c,\kappa_A\right)}{\partial c}=\frac{\kappa_A(1+e^{-\kappa_A c})}{(1+e^{-\kappa_A c})^2}. \tag{36}$$

Fig. 10 is showing the behavior of the sigmoid function and it's derivative for a variation of $\kappa_A$.

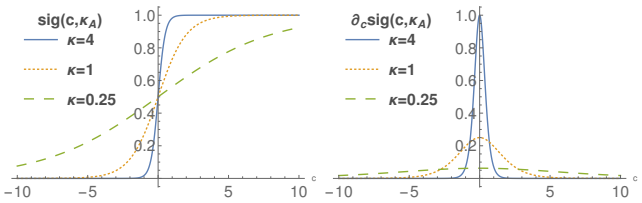


**Fig. 10** Sigmoid approximation of unit step with corresponding derivatives

For increasing $\kappa_A$, $sig\left(c,\kappa_A\right)$ is converging towards a step function $\epsilon(c)$. Its derivative is accordingly converging towards a $\delta$ impulse

$$\lim_{\kappa_A\to\infty} sig\left(c,\kappa_A\right)\to\epsilon(c) \tag{37}$$

$$\lim_{\kappa_A\to\infty}\frac{\partial sig\left(c,\kappa_A\right)}{\partial c}\to\delta(c). \tag{38}$$

Factor $\kappa_A$ is therefore determining the trade-off between quality of the approximation and condition number of the function. A bad-conditioned problem for high $\kappa_A$ is numerically more difficult to solve. The transformation of the extended weakened cone constraint (34) by means of a sigmoid function (35), finally yields to

$$\begin{aligned} &l_c\left(\mathbf{p}_{\mathcal{S}},\alpha,\kappa_G,\kappa_A\right) \\ &=sig\left(-x_{p\mathcal{S}},\kappa_A\right)\left(1+\kappa_G\left(x_{p\mathcal{S}}^2+y_{p\mathcal{S}}^2+z_{p\mathcal{S}}^2\right)\right) \\ &+sig\left(y_{p\mathcal{S}}^2+z_{p\mathcal{S}}^2-\left(x_{p\mathcal{S}}\sin(\alpha)\right)^2,\kappa_A\right) \\ &\cdot\ sig\left(x_{p\mathcal{S}},\kappa_A\right)\left(1+\kappa_G\left(y_{p\mathcal{S}}^2+z_{p\mathcal{S}}^2\right)\right) \end{aligned} \tag{39}$$

which is shown in Fig. 11 for $\kappa_A=10$.

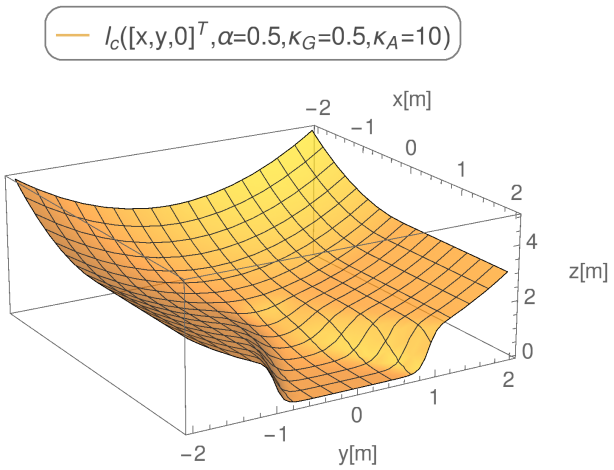


**Fig. 11** Extended weakened cone constraint transformed with sigmoid

To be able to track an object with known position in the global coordinate system, $\mathbf{p}_{\mathcal{S}}$ in (30) has to be determined by its counterpart $\mathbf{p}_{\mathcal{G}}$ in the global coordinate system. The required coordinate transformation is explained in the following section.

### 5.5 Coordinate Transformation

The coordinate transformation from the global coordinate frame to the sensor frame can be described as a sequential transformation with homogeneous transformation matrices. For a matrix that transforms a point $\mathbf{p}$ from the global coordinates $\mathcal{G}$ into the sensor frame coordinates $\mathcal{S}$ we use the following nomenclature

$$\begin{bmatrix}\mathbf{p}_{\mathcal{S}}\\1\end{bmatrix}=\mathbf{T}_{\mathcal{G}}^{\mathcal{S}}\begin{bmatrix}\mathbf{p}_{\mathcal{G}}\\1\end{bmatrix}. \tag{40}$$

This transformation matrix can consist of the position displacement of coordinate systems

$$\mathbf{T}\left(\mathbf{x}\right)=\begin{bmatrix}1&0&0&x\\0&1&0&y\\0&0&1&z\\0&0&0&1\end{bmatrix}\qquad\text{with }\mathbf{x}=\begin{bmatrix}x\ y\ z\end{bmatrix}^{\top}, \tag{41}$$

the rotational displacement expressed by an angle $\beta$ e.g. around $y$

$$\mathbf{T}_y(\beta)=\begin{bmatrix}\cos(\beta)&0&-\sin(\beta)&0\\0&1&0&0\\\sin(\beta)&0&\cos(\beta)&0\\0&0&0&1\end{bmatrix}, \tag{42}$$

and a direction vector e.g. around $z$

$$\mathbf{T}_z(\mathbf{d})=\begin{bmatrix}d_x&d_y&0&0\\-d_y&d_x&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}. \tag{43}$$

With the

$$\begin{aligned}&\text{robot position}&&\mathbf{p}_{r\mathcal{G}}=(x_{r\mathcal{G}},y_{r\mathcal{G}},y_{r\mathcal{G}})^{\top}\text{ and}\\&\text{robot orientation}&&\mathbf{d}_{r\mathcal{G}}=(d_{x\mathcal{G}},d_{y\mathcal{G}})^{\top}\end{aligned}$$

the transformation matrix from the global frame $\mathcal{G}$ into the vehicle frame $\mathcal{V}$

$$\begin{aligned}&\mathbf{T}_{\mathcal{G}}^{\mathcal{V}}\left(\mathbf{d}_{r\mathcal{G}},\mathbf{p}_{r\mathcal{G}}\right)=\mathbf{T}_{z_{\mathcal{G}}}(\mathbf{d}_{r\mathcal{G}})\mathbf{T}\left(-\mathbf{p}_{r\mathcal{G}}\right)\\&=\begin{bmatrix}d_{x\mathcal{G}}&d_{y\mathcal{G}}&0&-d_{x\mathcal{G}}x_{r\mathcal{G}}-d_{y\mathcal{G}}y_{r\mathcal{G}}\\-d_{y\mathcal{G}}&d_{x\mathcal{G}}&0&d_{y\mathcal{G}}x_{r\mathcal{G}}-d_{x\mathcal{G}}y_{r\mathcal{G}}\\0&0&1&-z_{r\mathcal{G}}\\0&0&0&1\end{bmatrix}\end{aligned} \tag{44}$$

and then to the sensor frame $\mathcal{S}$

$$\begin{aligned}&\mathbf{T}_{\mathcal{V}}^{\mathcal{S}}\left(\beta,\left[d_s,0,0\right]^{\top}\right)=\mathbf{T}_{y_{\mathcal{V}}}(\beta)\mathbf{T}\left(-\left[d_s,0,0\right]^{\top}\right)\\&=\begin{bmatrix}\cos(\beta)&0&-\sin(\beta)&-d_s\cos(\beta)\\0&1&0&0\\\sin(\beta)&0&\cos(\beta)&-d_s\sin(\beta)\\0&0&0&1\end{bmatrix}\end{aligned} \tag{45}$$

leads to the transformation matrix

$$\begin{aligned}&\mathbf{T}_{\mathcal{G}}^{\mathcal{S}}\left(\beta, d_s, \mathbf{d}_{r\mathcal{G}}, \mathbf{p}_{r\mathcal{G}}\right) \\ &= \mathbf{T}_{\mathcal{V}}^{\mathcal{S}}\left(\beta, \left[d_s, 0, 0\right]^{\top}\right) \mathbf{T}_{\mathcal{G}}^{\mathcal{V}}\left(\mathbf{d}_{r\mathcal{G}}, \mathbf{p}_{r\mathcal{G}}\right) \\ &= \begin{bmatrix} d_{x\mathcal{G}}\cos(\beta) & d_{y\mathcal{G}}\cos(\beta) & -\sin(\beta) & z_{r\mathcal{G}}\sin(\beta) - \nu\cos(\beta) \\ -d_{y\mathcal{G}} & d_{x\mathcal{G}} & 0 & d_{y\mathcal{G}}x_{r\mathcal{G}} - dxy_{r\mathcal{G}} \\ d_{x\mathcal{G}}\sin(\beta) & d_{y\mathcal{G}}\sin(\beta) & \cos(\beta) & -z_{r\mathcal{G}}\cos(\beta) - \nu\sin(\beta) \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &\text{with } \nu = d_s + d_{x\mathcal{G}}x_{r\mathcal{G}} + d_{y\mathcal{G}}y_{r\mathcal{G}},\end{aligned} \tag{46}$$

and the inverse transformation

$$\begin{aligned}&\mathbf{T}_{\mathcal{S}}^{\mathcal{G}}\left(\beta, d_s, \mathbf{d}_{r\mathcal{G}}, \mathbf{p}_{r\mathcal{G}}\right) \\ &= \left(\mathbf{T}_{\mathcal{G}}^{\mathcal{S}}\left(\beta, d_s, \mathbf{d}_{r\mathcal{G}}, \mathbf{p}_{r\mathcal{G}}\right)\right)^{-1} \\ &= \begin{bmatrix} d_{x\mathcal{G}}\cos(\beta) & -d_{y\mathcal{G}} & d_{x\mathcal{G}}\sin(\beta) & d_s d_{x\mathcal{G}} + x_{r\mathcal{G}} \\ d_{y\mathcal{G}}\cos(\beta) & d_{x\mathcal{G}} & d_{y\mathcal{G}}\sin(\beta) & d_s d_{y\mathcal{G}} + y_{r\mathcal{G}} \\ -\sin(\beta) & 0 & \cos(\beta) & z_{r\mathcal{G}} \\ 0 & 0 & 0 & 1 \end{bmatrix}.\end{aligned} \tag{47}$$

The global coordinates in the sensor frame is accordingly given by (40) and leads to

$$\begin{aligned}&\begin{bmatrix}\mathbf{p}_{\mathcal{S}} \\ 1\end{bmatrix} = \mathbf{T}_{\mathcal{G}}^{\mathcal{S}}\left(\beta, d_s, \mathbf{d}_{r\mathcal{G}}, \mathbf{p}_{r\mathcal{G}}\right)\begin{bmatrix}\mathbf{p}_{\mathcal{G}} \\ 1\end{bmatrix} \\ &= \begin{bmatrix} x_{p\mathcal{S}} \\ y_{p\mathcal{S}} \\ z_{p\mathcal{S}} \\ 1 \end{bmatrix} = \begin{bmatrix} (z_{r\mathcal{G}} - z_{p\mathcal{G}})\sin(\beta) - \eta\cos(\beta) \\ d_{y\mathcal{G}}(x_{r\mathcal{G}} - x_{p\mathcal{G}}) + d_{x\mathcal{G}}(y_{p\mathcal{G}} - y_{r\mathcal{G}}) \\ (z_{p\mathcal{G}} - z_{r\mathcal{G}})\cos(\beta) - \eta\sin(\beta) \\ 1 \end{bmatrix} \\ &\text{with } \eta = d_s + d_{x\mathcal{G}}\left(x_{r\mathcal{G}} - x_{p\mathcal{G}}\right) + d_{y\mathcal{G}}\left(y_{r\mathcal{G}} - y_{p\mathcal{G}}\right)\end{aligned} \tag{48}$$

Finally the target position in the sensor coordinate frame can be expressed in global coordinates by (48) and applied in $l_c\left(\mathbf{p}_{\mathcal{S}}, \alpha, \kappa_G, \kappa_A\right)$ (39). Due to its complexity, the resulting equation is not given here. Fig. 12 is showing the resulting costs in the $xy$-plane for a quadrotor at position $\mathbf{p}_{r\mathcal{G}} = (-1, -1, 0)^{\top}$ and orientation $\mathbf{d}_{r\mathcal{G}} = (0.71, 0.71)^{\top} \equiv \Psi = 45°$. The triangular base form of the cone is oriented as expected from the $UAV$ origin in $\mathbf{p}_{r\mathcal{G}} = (-1, -1, 0)^{\top}$.

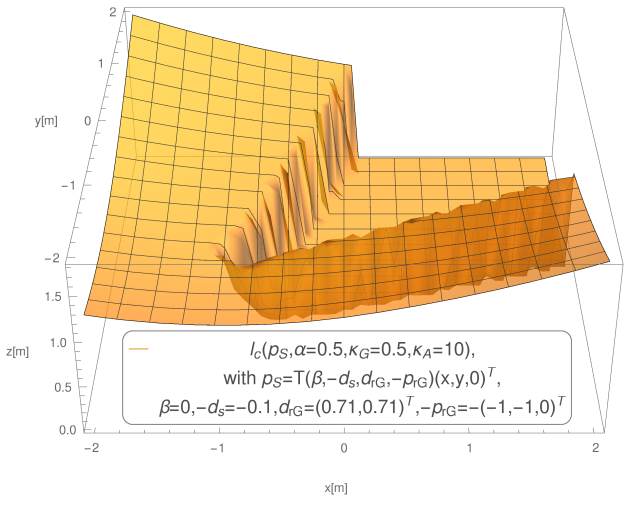


**Fig. 12** Tracking of global position with weakened cone constraint

## 6 Safety Constraints

To validate the derived potential function experimentally, additional safety measures are necessary. The most important safety constraint is treating collision avoidance and ensures that a safety distance $d_{min}$ between object and drone is not violated. Second, the maximum distance $d_{max}$ of the sensor has to be considered. This can be accomplished by implementing a cohesion constraint which introduces a repulsive behavior from large distances between object and robot. Both constraints are derived according to the workflow presented for the sensor constraint in section 5.

### 6.1 Collision Avoidance Constraint

Considering that $\mathbf{p}_{\mathcal{G}}$ could also represent an obstacle to avoid, a potential function for obstacle avoidance can be formulated as penalty of a distance below a limit $d_{min}$:

$$d_{min} \leq \left\|\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}}\right\|. \tag{49}$$

The norm can be reformulated by means of quadrature

$$0 \leq c_{minD} = \left(\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}}\right)^{\top}\left(\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}}\right) - d_{min}^2 \tag{50}$$

which then is transformed into a potential function with unit steps

$$\begin{aligned}l_{minD} &= \epsilon\left(c_{minD}\right) \\ &= \epsilon\left(\left(\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}}\right)^{\top}\left(\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}}\right) - d_{min}^2\right).\end{aligned} \tag{51}$$

To improve the convergence properties, (51) is extended by a quadratic penalty term

$$\begin{aligned}l_{minD} = \epsilon&\left(\left(\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}}\right)^{\top}\left(\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}}\right) - d_{min}^2\right) \\ &\cdot \left(\kappa_H - \kappa_G\left(\left(\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}}\right)^{\top}\left(\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}}\right)\right)\right)\end{aligned} \tag{52}$$

Here, $\kappa_H$ is defining the maximum height and $\kappa_G$ the decent of the gradient of the potential function. Finally, (52) can be transformed to an analytical function with the help of the sigmoid function (36)

$$l_{minD}\left(\mathbf{p}_{\mathcal{G}}, \mathbf{p}_{r\mathcal{G}}, d_{min}, \kappa_H, \kappa_G, \kappa_A\right) \tag{53}$$

$$\begin{aligned}= sig&\left(\left(\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}}\right)^{\top}\left(\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}}\right) - d_{min}^2, \kappa_A\right) \\ &\cdot \left(\kappa_H - \kappa_G\left(\left(\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}}\right)^{\top}\left(\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}}\right)\right)\right).\end{aligned} \tag{54}$$

$\kappa_A$ is describing the quality of the unit step approximation as before. The final result of the potential function is shown in Fig. 13 which shows a high penalty for the area with distance $d_{min}$ around the origin. The form of the convex top can be adjusted with $\kappa_G$. Fig. 14 is showing penalty values $l_{minD} > 0.5$ for an obstacle in position $\mathbf{p}_{\mathcal{G}} = (x, y, z)^{\top}$ with robot placed at $\mathbf{p}_{r\mathcal{G}} = (-1, -1, 0)^{\top}$. As desired any violation of $d_{min}$ (49) is addressed with a high penalty. This yields a repulsive behavior between robot and object.

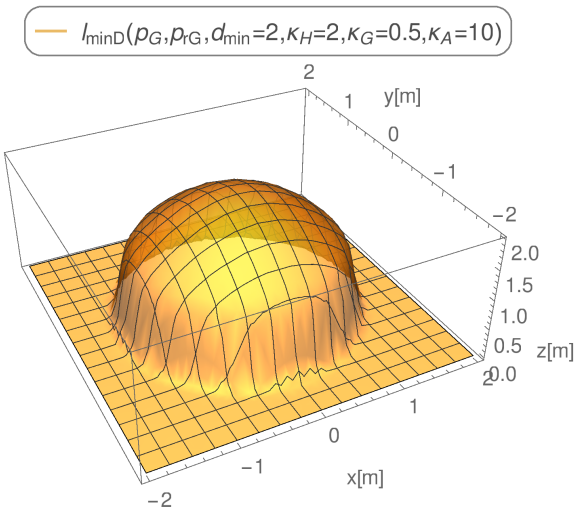


**Fig. 13** Collision avoidance constraint potential function over $xy$-plane

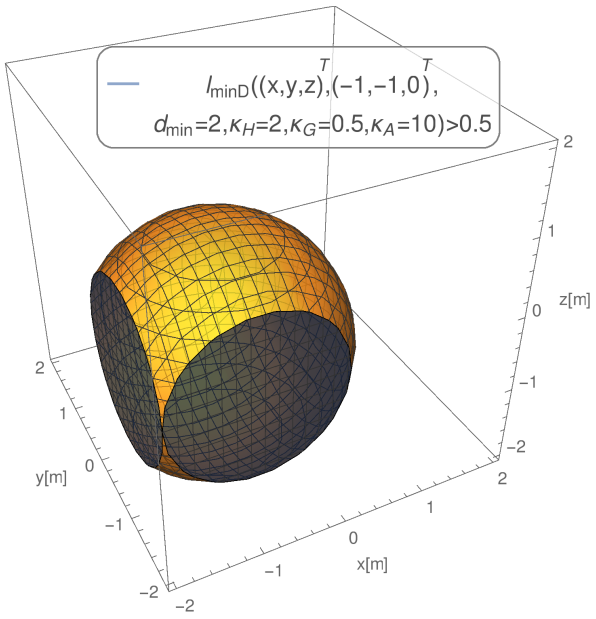


**Fig. 14** Collision avoidance constraint potential function in global perspective

## 6.2 Cohesion Constraint

Cohesion can be seen as inversion of the collision avoidance problem (49), where distances bigger than $d_{max}$ to an object should be avoided. The constraint can therefore be formulated as

$$d_{max} \geq \|\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}}\| \,. \tag{55}$$

As before, the norm can be expressed with a the help of a quadrature

$$0 \leq c_{maxD} = d_{max}^2 - (\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}})^{\top} (\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}}) \tag{56}$$

and transformed into a cost function with a unit step function

$$\begin{aligned} l_{maxD} &= \epsilon\left(c_{maxD}\right) \\ &= \epsilon\left(d_{max}^2 - (\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}})^{\top} (\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}})\right). \end{aligned} \tag{57}$$

To manipulate the curvature of the given penalty function, $\kappa_H$ is introduced to define the maximum height. $\kappa_G$ is describing the ascent of the gradient of the potential function

$$\begin{aligned} l_{maxD} = \epsilon &\left(d_{max^2} - (\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}})^{\top} (\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}})\right) \\ \cdot &\left(\kappa_H + \kappa_G \left((\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}})^{\top} (\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}})\right)\right) \end{aligned} \tag{58}$$

The final analytical cost function is gained from an approximation of the unit step $\epsilon$ with the sigmoid function (36). Which yields

$$\begin{aligned} &l_{maxD}\left(\mathbf{p}_{\mathcal{G}}, \mathbf{p}_{r\mathcal{G}}, d_{max}, \kappa_H, \kappa_G, \kappa_A\right) \\ &= sig\left(d_{max}^2 - (\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}})^{\top} (\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}}), \kappa_A\right) \\ &\cdot \left(\kappa_H + \kappa_G \left((\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}})^{\top} (\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}})\right)\right). \end{aligned} \tag{59}$$

Fig. 15 is showing the desired inverse behavior as for the collision avoidance behavior in Fig. 13. The cohesion constraint leads to a high penalty for distances greater than $d_{max}$ around the origin. Fig. 14 is showing penalty values $l_{maxD} > 0.5$ for an obstacle in position $\mathbf{p}_{\mathcal{G}} = (x, y, z)^{\top}$ with the robot placed at $\mathbf{p}_{r\mathcal{G}} = (-1, -1, 0)^{\top}$. The figure validates that the whole area of $\|\mathbf{p}_{\mathcal{G}} - \mathbf{p}_{r\mathcal{G}}\| \geq d_{max}$ is highly penalized which leads to an attracting behavior between object and robot.

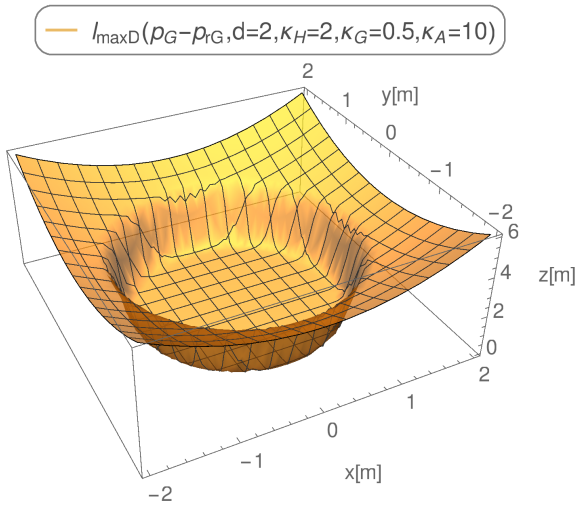


**Fig. 15** Cohesion constraint function over $xy$-plane

With the developed safety constraints (53) and (59), the cone constraint has been validated as discussed in the following section.

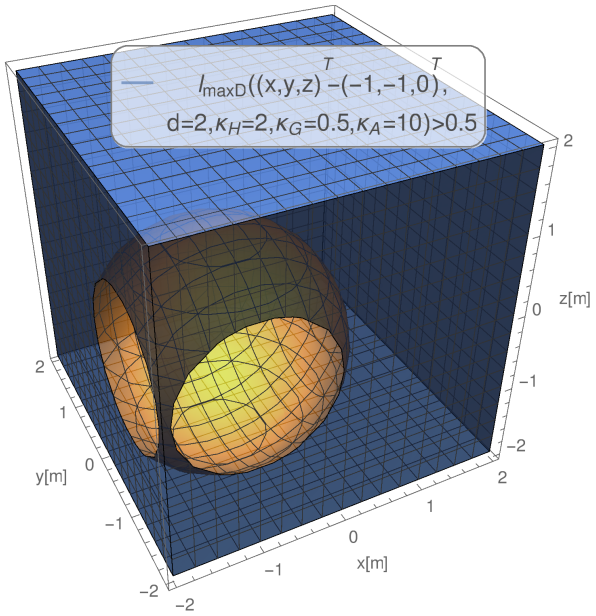


**Fig. 16** Cohesion constraint potential function in global perspective

## 7 Validation of the Sensor Based Control With Potential Functions

The validation scenario is a visual quadrotor tracking scenario, where one quadrotor is equipped with a camera and tracking a target quadrotor. The task is to keep the target quadrotor in the camera frame as shown in Fig. 17.

**Fig. 17** Visual tracking of quadrotor from camera-equipped quadrotor

The control of both quadrotors is accomplished with a central $MPC$ controller [49] and extended by the potential functions that are defining the tracking task. The combination of cone (39), collision avoidance (53) and cohesion (59) constraint with the coordinate transformation (48) and quadrotor dynamics (14) with parameters (16) in an $OCP$ results in

$$\begin{aligned}
\min_{\mathbf{u}} \; J = &\int_{t_0}^{t_f} l_0(\tau) + l_1(\tau) \\
&+k_{00} l_c^{\setminus\{u_1\}}(\mathbf{p}_{uav1\mathcal{S}}, \alpha, \kappa_{G0}, \kappa_{A0}) \\
&+k_{01} l_{maxD}^{\setminus\{u_1\}}(\mathbf{p}_{uav1\mathcal{G}}, \mathbf{p}_{uav0\mathcal{G}}, d_{max}, \kappa_{H1}, \kappa_{G1}, \kappa_{A1}) \\
&+k_{02} l_{minD}^{\setminus\{u_1\}}(\mathbf{p}_{uav1\mathcal{G}}, \mathbf{p}_{uav0\mathcal{G}}, d_{min}, \kappa_{H2}, \kappa_{G2}, \kappa_{A2})\, \mathrm{d}\tau \\
\text{s.t. } &\mathbf{c}_0(\tau) \leq \mathbf{0}, \mathbf{c}_1(\tau) \leq \mathbf{0} \\
&\mathbf{0} = \mathbf{f}_0(\mathbf{x}_0, \mathbf{u}_0, t), \mathbf{0} = \mathbf{f}_1(\mathbf{x}_1, \mathbf{u}_1, t)
\end{aligned} \tag{60}$$

with

$$\begin{bmatrix} \mathbf{p}_{uav1\mathcal{S}} \\ 1 \end{bmatrix} = \mathbf{T}_{\mathcal{G}}^{\mathcal{S}}(\beta, -d_s, \mathbf{d}_{uav0\mathcal{G}}, -\mathbf{p}_{uav0\mathcal{G}}) \begin{bmatrix} \mathbf{p}_{uav1\mathcal{G}} \\ 1 \end{bmatrix}$$

$$\begin{aligned}
l_i(\tau) = &(\mathbf{x}_i^*(\tau) - \mathbf{x}_i(\tau))^\top \mathbf{Q}_i (\mathbf{x}_i^*(\tau) - \mathbf{x}_i(\tau)) \\
&+\mathbf{u}_i(\tau) \mathbf{R}_i \mathbf{u}_i(\tau)
\end{aligned}$$

The index $\setminus\{u_1\}$ of the cost functions $l_c$, $l_{maxD}$ and $l_{minD}$ reflects, that the influence of the cost functions on $UAV_1$ is neglected, as explained in (10). This means that the cone, collision avoidance and cohesion constraint is only affecting $UAV_0$. The advantage of this modular approach of a central control for such a scenario is, that future states of both systems are considered for the computation of the optimal controls, while the effect of the tracking costs can be limited to one quadrotor.

The parameters of (60) for the experimental validation are given as

$$\mathbf{Q}_0 = Diag\left(\left[0,0,0,0,0,0.7,0.7\right]\right) \tag{61}$$

$$\mathbf{R}_0 = Diag\left(\left[1,1,1,1\right]\right) \tag{62}$$

$$\mathbf{Q}_1 = Diag\left(\left[1.5,1.5,1.6,1,1,0,0\right]\right) \tag{63}$$

$$\mathbf{R}_1 = Diag\left(\left[1,1,1,1\right]\right) \tag{64}$$

$$\begin{aligned}
p_c \;:\; &d_s = 0.17, \alpha = 0.5, \beta = 0.5, \\
&k_{00} = 0.4, k_{G0} = 0.01, k_{A0} = 2.0
\end{aligned} \tag{65}$$

$$\begin{aligned}
p_{minD} \;:\; &d_{min} = 1, k_{01} = 0.4, \\
&k_{H1} = 4.5, k_{G1} = 0.001, k_{A1} = 3.0
\end{aligned} \tag{66}$$

$$\begin{aligned}
p_{maxD} : &d_{max} = 2, k_{02} = 0.4, \\
&k_{H2} = 1.5, k_{G2} = 0.001, k_{A2} = 3.0
\end{aligned} \tag{67}$$

$$\begin{aligned}
p_{cgmres} : &n_{hor} = 20, T_{hor} = 1\mathrm{s}, \epsilon_T = 10^{-8}, \zeta = 10 \\
&\Delta t = 0.01\mathrm{s}, k_{max} = 30, \alpha_{hor} = 2
\end{aligned} \tag{68}$$

The parameters $p_{cgmres}$ are referring to the $CMSCGMRES$ solver and are given in a coherent notation to the previous work [2]. To examine the dynamic behavior of the proposed control solution, the target position of $UAV_1$ is moving in a circular trajectory. For $UAV_0$, the choice of $\mathbf{Q}_0$ leads to a tracking of zero forward velocity $\dot{x}_{uav0\mathcal{V}}(t) = 0$ and sideward velocity $\dot{y}_{uav0\mathcal{V}}(t) = 0$ which yields to the desired states

$$\mathbf{x}_0^* = Diag\left(\left[0,0,0,0,0,0.0,0.0\right]\right) \tag{69}$$

$$\mathbf{x}_1^* = Diag\left(\left[\tfrac{1}{2}\cos(0.3t), \tfrac{1}{2}\sin(0.3t), 1, 0, 0, 0.0, 0.0\right]\right) \tag{70}$$

Based on the given parameters and target trajectories, the scenario is validated in the following sections.

### 7.1 Numerical Validation

For the numerical validation, two $AR.Drone\,2.0$ models have been implemented in the simulation environment $V\text{-}REP$ (Fig. 18). The position information of quadrotors and target trajectories is shared via a Robot Operating System ($ROS$) interface.

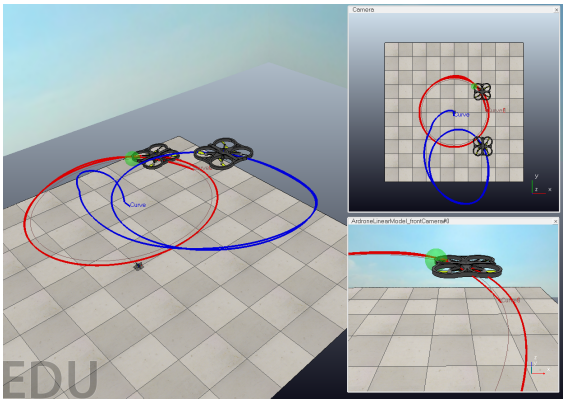

**Fig. 18** Numerical validation of the use case scenario with $V\text{-}REP$

Fig. 19 is showing the trajectories of both $UAV$s. $UAV_1$ is following the circular moving target which is reflected by sinusoidal position trajectories. $UAV_0$ is tracking $UAV_1$ with the developed sensor constraints which is indicated by likewise sinusoidal position trajectories of $UAV_0$. The form and position of the resulting trajectory is depending on the initial $UAV$ positions. As the position of $UAV_0$ is just dependent on the applied constraints, $UAV_0$ can rotate freely around $UAV_1$. This explains the drift of the sinusoidal position trajectory of $UAV_0$. The control trajectories of $UAV_1$ and $UAV_0$ are showing that the input limits are respected. The distance $d$ stays in the defined limits $d_{min} \leq d \leq d_{max}$ with one exception at the initial phase of the simulation. Here the repulsive behavior of the collision avoidance constraint can be seen, where the controller increases the distance to fulfill $d \geq d_{min}$. On one hand, the disadvantage of the weakened constraint is, that $d$ can violate the given constraint depending on the parametrization of the potential function and the smoothness of the unit step approximation by the sigmoid (35). On the other hand, the advantage is, that a constraint violation does not lead to an infeasible $OCP$. To conclude, the distance trajectory is validating the active collision avoidance and cohesion constraint.

To validate that the tracked $UAV_1$ stays within the sensor beam width angle $\alpha$, Fig. 19 is therefore also showing the absolute tracking angle $\alpha_t$

$$\alpha_t = \| \arccos(\frac{\mathbf{p}_\mathcal{S} \cdot (1,0,0)^\top}{|\mathbf{p}_\mathcal{S}|}) \|. \tag{71}$$

which is describing the angle between the $UAV_0$ sensor orientation vector and the distance vector to $UAV_1$ in the sensor frame. The resulting $\alpha_t$ plot in Fig. 19 is validating, that

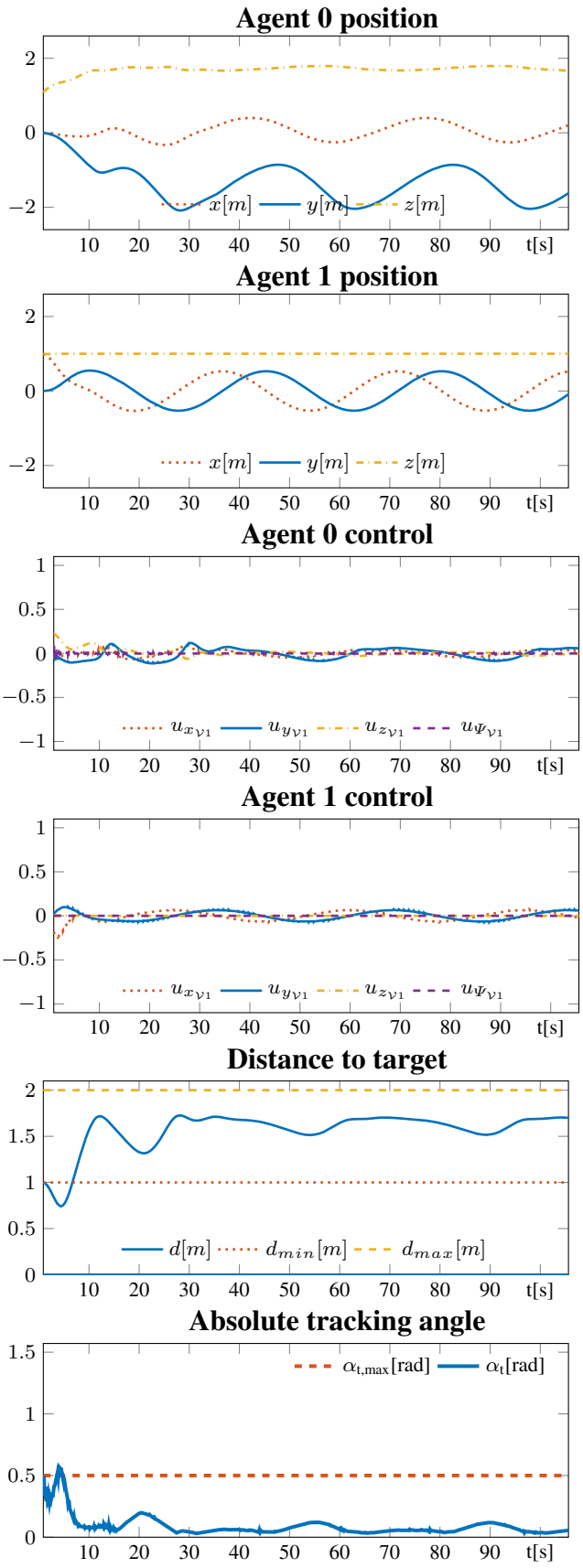


**Fig. 19** Numerical simulation: Trajectories of use case scenario

the cone constraint keeps the tracking angle $\alpha_t$ smaller than the sensor beam width angle $\alpha_t \leq \alpha = 0.5\text{rad}$. It can be seen that the tracking angle $\alpha_t$ is in fact much smaller. This is caused by the smooth approximation of the unit steps by sigmoids (35) which leads to a convergence towards the center of the cone. Especially for undisturbed systems, this yields to a smaller tracking angle $\alpha_t$ than the beam width angle $\alpha$. To reduce this effect $\kappa_A$ can be increased.

A more intuitive access to the $UAV$ behavior is gained by plotting the $UAV$ positions and the orientation of $UAV_0$ by means of a vector as shown in Fig. 20. To be able to associate both $UAV$ positions, time-related $UAV$ positions are connected with a line. It is visible that $UAV_1$ is following the desired circular trajectory, while $UAV_0$ is tracking $UAV_1$ in an ellipsoidal movement. The orientation vectors are displayed at each $\Delta t \approx 1.68\text{s}$ for means of visualization.

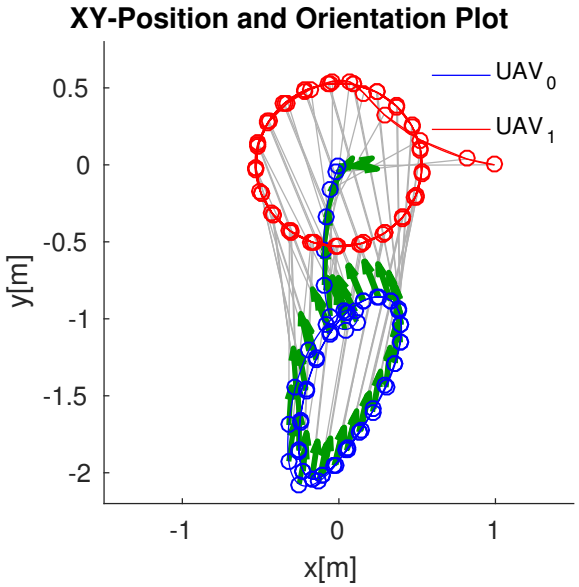


**Fig. 20** $xy$-plot with $UAV_0$ orientation of the numerical simulation of the use case scenario (Direction samples each $\Delta t \approx 1.67\text{s}$)

To resume, Fig. 19-20 validate the desired behavior of the proposed sensor constrained $MPC$ controller in simulation. The next section is discussing the extension of this numerical validation to the real scenario.

## 7.2 Experimental Validation

The use case scenario with real $AR.Drone\,2.0$ quadrotors, as shown in Fig. 21, is subject to a variety of disturbance which is not considered in the numerical validation. Lightweight $UAV$s like an $AR.Drone\,2.0$ quadrotor are particularly responsive to airflow disturbance. In addition, their flight dynamics are very volatile, as their body consist of deformable Styrofoam. These and more influences (communi-

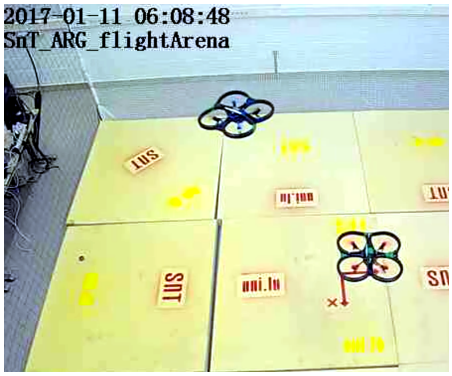


**Fig. 21** Experimental validation of the use case scenario with real quadrotors

cation latency, prediction model errors, measurement uncertainty) are not considered in the $MPC$ model and simulation environment. For this reason an experimental validation is necessary to assess the robustness of the proposed control approach. The position data of the real $AR.Drone\,2.0$ quadrotors is hereby measured by a motion capture system.

The resulting system trajectories are given in Fig. 22. In contrast to the numerical simulation, the real $AR.Drone\,2.0$ trajectories are subject to significant disturbance. Especially the mutual airflow disturbance is causing low-frequency oscillations $\approx 0.5\text{Hz}$ which are visible in the $y$-position of $UAV_0$ and $UAV_1$. As the airflow disturbance error is propagating from the $UAV_1$ position to the $UAV_0$ response, the resulting oscillations are particularly visible in the controls of $UAV_0$.

To evaluate the control performance, the distance $d$ and absolute tracking angle $\alpha_t$ are given in Fig. 22. The distance plot is stating that $UAV_1$ is tracked within the given distance limitations. At $t \approx 55\text{s}$ the behavior of the weakened constraint is visible which is allowing a minor violation of $d_{min}$ in return for avoiding infeasible solutions and ease of implementation. The absolute tracking angle (71) in Fig. 22 is measured between the $UAV_0$ orientation vector and the distance vector to $UAV_1$. Due to the initial conditions, $\alpha_t$ is violating the applied cone constraint at the beginning, but is tracked within the given beam width angle $\alpha_t \leq \alpha = 0.5\text{rad}$ for $t \geq 6\text{s}$. Hence, the trajectories in Fig. 22 confirm the desired tracking behavior.

Fig. 23 is showing the resulting $xy$-trajectory with samples of the $UAV_0$ orientation vector. For means of visualization, the orientation sample time is reduced to $\Delta t \approx 1.68\text{s}$. Due to different initial conditions (positions, velocities, etc.), the $xy$-trajectory is not directly comparable with the numerical simulation. However, it visualizes the closed-loop $UAV$ position response to mutual airflow disturbance in the form of small oscillation. In comparison to the simulation trajectory, the rejection of these oscillations (input costs) dominate the minor gradient within the compliant area of the sensor constraint. As a result the absolute tracking angle $\alpha_t$ and distance $d$ plots in Fig. 22 go closer to their

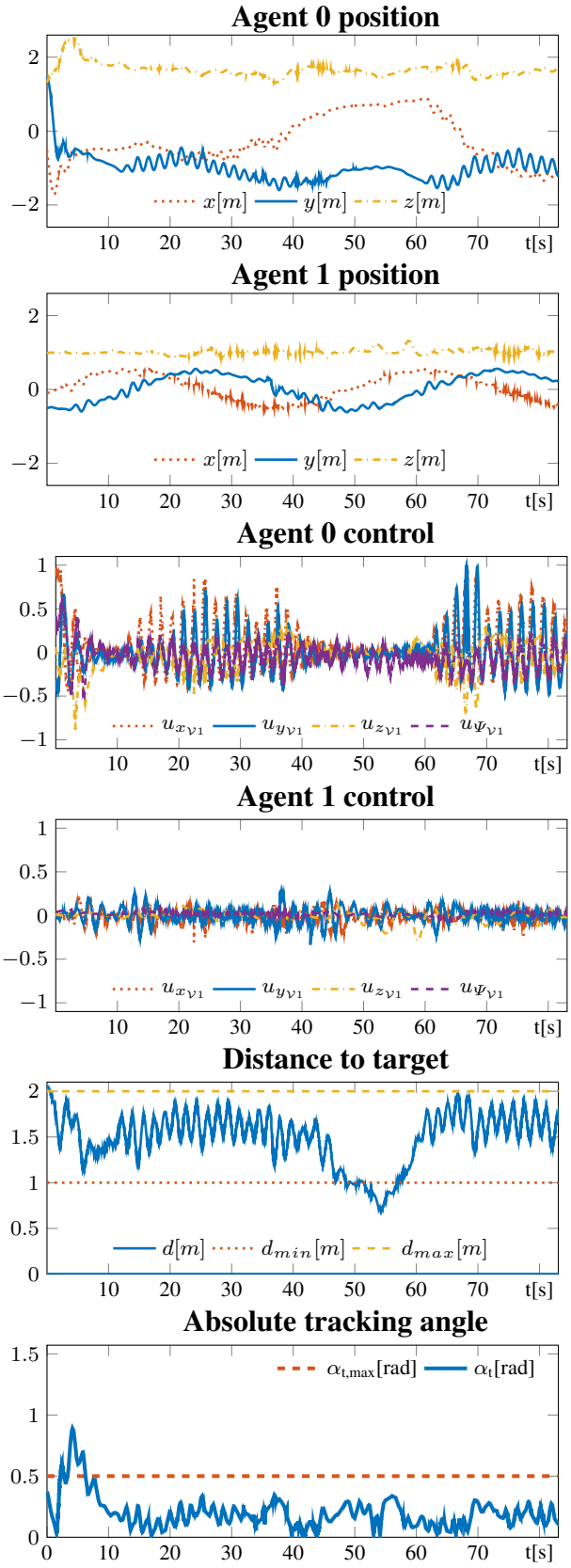


**Fig. 22** Trajectories of real use case scenario

constraint limits and the $xy$-plot does not follow the ellipsoidal pattern of $UAV_0$ in 20. In order to evaluate the tracking without the mutual airflow disturbance and under different initial conditions, a further analysis of the robustness is shown in the experimental discussion (section 7.3).

For the evaluation of the computational efficiency of the proposed approach, Fig. 24 is giving the $MPC$ computation time on a standard computer ($Dell\ Latitude\ E5440$). The peaks in the computation time of $t_{comp} \approx 20\text{ms}$ are not directly related to the solver, but are caused by CPU interrupts by other processes. The resulting average computation time $t_{av,comp} = 2.5\text{ms}$ is very low in comparison to the control update interval of $\Delta t = 10\text{ms}$ and states the real-time feasibility of the $MPC$.

To conclude, the experimental results are validating the computational efficiency and effectiveness of the $MPC$ control based on potential functions.

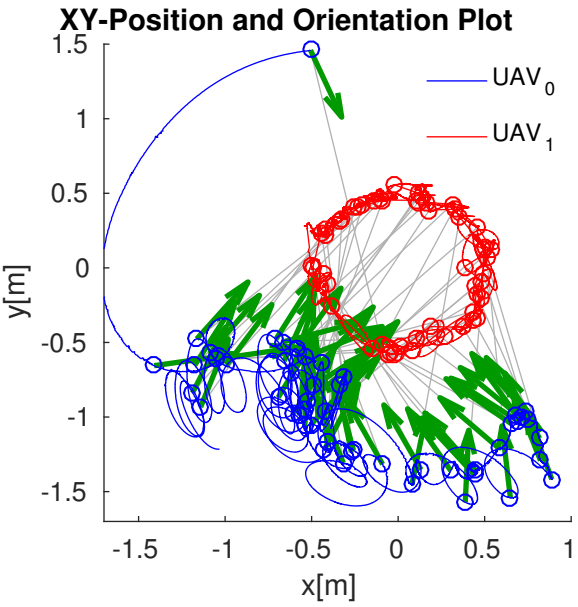


**Fig. 23** $xy$-plot with $UAV_0$ orientation of real use case scenario (Direction samples each $\Delta t \approx 1.68\text{s}$)

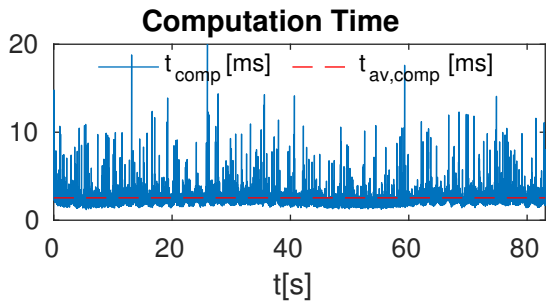


**Fig. 24** Computation time of real use case scenario

### 7.3 Experimental Discussion

In order to discuss the robustness to disturbance and initial conditions, the previous experiment of section 7.2 is altered by substituting $UAV_1$ by a target with constant position $\mathbf{p}_{uav0\mathcal{G}} \rightarrow \mathbf{p}_{t\mathcal{G}} = [0.04, -0.05, 0.96]^\top \,\mathrm{m}$ . The disturbance is introduced manually by means of an obstacle with the position $\mathbf{p}_{O\mathcal{G}}$ and a related collision avoidance constraint (52) $l_{minD}\left(\mathbf{p}_{O\mathcal{G}}, \mathbf{p}_{uav0\mathcal{G}}, d_{O,min}, \kappa_H 3, \kappa_G 3, \kappa_A 3\right)$. In order to make the evasive behavior visible in the $xy$-plane, the $z$-axis action of $UAV_0$ is reduced by increasing the input penalty for $u_z$. The resulting $OCP$ (72)-(78) does consider the obstacle and target position within the prediction horizon as constant

$$
\begin{aligned}
\min_{\mathbf{u}} \; J = & \int_{t_0}^{t_f} \mathbf{u}_0(\tau)\, \mathbf{R}_0 \mathbf{u}_0(\tau) \qquad (72)\\
& + k_{00} l_c\left(\mathbf{p}_{t\mathcal{S}}, \alpha, \kappa_{G0}, \kappa_{A0}\right)\\
& + k_{01} l_{maxD}\left(\mathbf{p}_{t\mathcal{G}}, \mathbf{p}_{uav0\mathcal{G}}, d_{max}, \kappa_{H1}, \kappa_{G1}, \kappa_{A1}\right)\\
& + k_{02} l_{minD}\left(\mathbf{p}_{t\mathcal{G}}, \mathbf{p}_{uav0\mathcal{G}}, d_{min}, \kappa_{H2}, \kappa_{G2}, \kappa_{A2}\right)\\
& + k_{03} l_{minD}\left(\mathbf{p}_{O\mathcal{G}}, \mathbf{p}_{uav0\mathcal{G}}, d_{O,min}, \kappa_{H3}, \kappa_{G3}, \kappa_{A3}\right) \mathrm{d}\tau\\
\text{s.t. } & \mathbf{c}_0(\tau) \leq \mathbf{0}, \mathbf{c}_1(\tau) \leq \mathbf{0}\\
& \mathbf{0} = \mathbf{f}_0\left(\mathbf{x}_0, \mathbf{u}_0, t\right)
\end{aligned}
$$

with

$$
\begin{bmatrix} \mathbf{p}_{t\mathcal{S}} \\ 1 \end{bmatrix} = \mathbf{T}_{\mathcal{G}}^{\mathcal{S}}\left(\beta, -d_s, \mathbf{d}_{uav0\mathcal{G}}, -\mathbf{p}_{uav0\mathcal{G}}\right) \begin{bmatrix} \mathbf{p}_{t\mathcal{G}} \\ 1 \end{bmatrix}
$$

$$
\mathbf{R}_0 = \mathit{Diag}\left(\left[1, 1, 10, 1\right]\right) \qquad (73)
$$

$$
\begin{aligned}
p_c &: d_s = 0.17, \alpha = 0.5, \beta = 0.5, && (74)\\
& \quad k_{00} = 0.4, k_{G0} = 0.01, k_{A0} = 2.0 \\
p_{minD} &: d_{min} = 0.7, k_{01} = 0.4, && (75)\\
& \quad k_{H1} = 4.5, k_{G1} = 0.0001, k_{A1} = 3.0 \\
p_{maxD} &: d_{max} = 2, k_{02} = 0.4, && (76)\\
& \quad k_{H2} = 1.5, k_{G2} = 0.0001, k_{A2} = 3.0 \\
p_{OminD} &: d_{O,min} = 1, k_{03} = 0.6, && (77)\\
& \quad k_{H3} = 1.5, k_{G3} = 0.001, k_{A3} = 3.0 \\
p_{cgmres} &: n_{hor} = 20, T_{hor} = 1\mathrm{s}, \epsilon_T = 10^{-8}, \zeta = 10 && (78)\\
& \quad \Delta t = 0.01\mathrm{s}, k_{max} = 30, \alpha_{hor} = 2.
\end{aligned}
$$

The experimental outline is shown in Fig. 25. As in section 7.2, the pose of $UAV_0$ is measured with a motion capture system. The constant target position is indicated as green diamond in the center of the pictures. In its initial pose, $UAV_0$ is not necessarily fulfilling the underlying inequality constraints of the sensor cone constraint (39). The activation of the controller therefore initially leads to a convergence towards a compliant pose. As a next step, an obstacle (red star) is introduced manually by means of the motion capture system. $UAV_0$ (blue circle) is evading the approaching obstacle by moving in the opposite direction. This evasion maneuver shows the form of an arc due to the active target tracking constraints. Following $UAV_0$ with the obstacle consequently leads to a circular trajectory around the tracked target point $\mathbf{p}_{t\mathcal{G}}$. The corresponding $UAV_0$ orientation is indicated by a blue arrow.

To cope different initial conditions and obstacle patterns, a set of 10 experiments is conducted. Fig. 26 is showing the resulting $xy$-trajectory of $UAV_0$ (blue line), obstacle (dashed red line) and the fixed target position (green diamond). To visualize the system at different time instances, the position of the obstacle (red star) and pose of $UAV_0$ (blue circle with arrow) is marked every $\Delta t = 4.2\mathrm{s}$. The distances between $UAV_0$ and target is signalized as light grey line at each of these time instances. Accordingly, the distance between $UAV_0$ and the obstacle is indicated as light red line. The resulting circular patters validates the target tracking during the evasion maneuver. Furthermore, the direction of the drone is pointing to the center which is indicating the tracking of the target. In this context, in plot 7 center left and plot 9 center up the drone orientations which are not pointing towards the target are representing initial conditions. These initial poses of $UAV_0$ have been chosen arbitrarily within the spacial limitations of the laboratory and its exact values can be exerted from the $UAV_0$ pose plots in Fig. 27. In this initial phase a typical increase in the altitude ($z$) can be observed which is caused by the inclination angle $\beta$ of the sensor cone constraint. The steps in the $\Psi$ values in Fig. 27 are based on the limited $\Psi$-angle interval. The sinusoidal trajectories in $x$ and $y$ evidence the circular evasion maneuver of $UAV_0$. Fig. 27 is showing measurements of the corresponding obstacle position. Also here the sinusoidal trajectory, in order to follow the $UAV_0$ and provoke an evasion, is visible. The steps in the obstacle position at the beginning are caused by entering the detection zone of the motion capture system.

To analyze the influence of the obstacle collision avoidance constraint $l_{minD}$ (53), Fig. 29 is showing the distance between $UAV_0$ and obstacle. The plots show how the obstacle is moved close to $d_{Omin}$ to provoke an evasion maneuver of $UAV_0$. The measured minimal distances are given in Table 7.3. As the collision avoidance design is based on weakened constraints, violations are feasible. For all 10 experiments the highest violation of the obstacle distance $d_O$ appears in run 5 with $\min(d_O) = 0.601\mathrm{m}$. The violation generally depends on $UAV_0$ and obstacle speed as well as cost gradient design of the collision avoidance constraint. For the here considered $UAV$ and obstacle speeds, the cost gradient is chosen less steep (77) in order to show a smooth repulsive behavior while showing the desired evasion. In reverse conclusion higher violations are accepted. For higher system velocities this cost gradient has to be chosen steeper. Its repulsive behavior can be observed as oscillation around the constraint border in Fig. 29 run 6, 7 and 9.

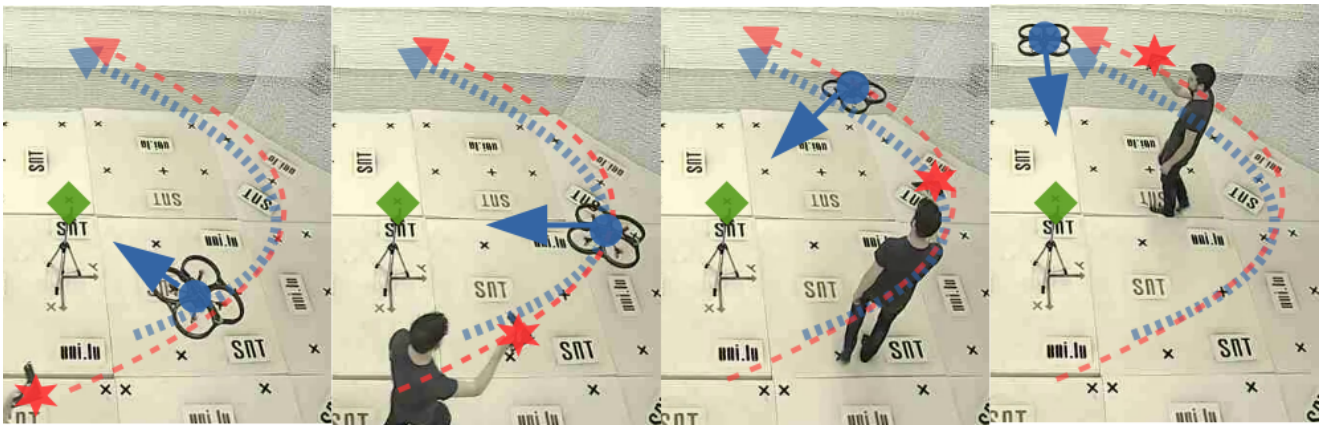

**Fig. 25** $UAV_0$ trajectory - tracking target with constant position while obstacle is avoided

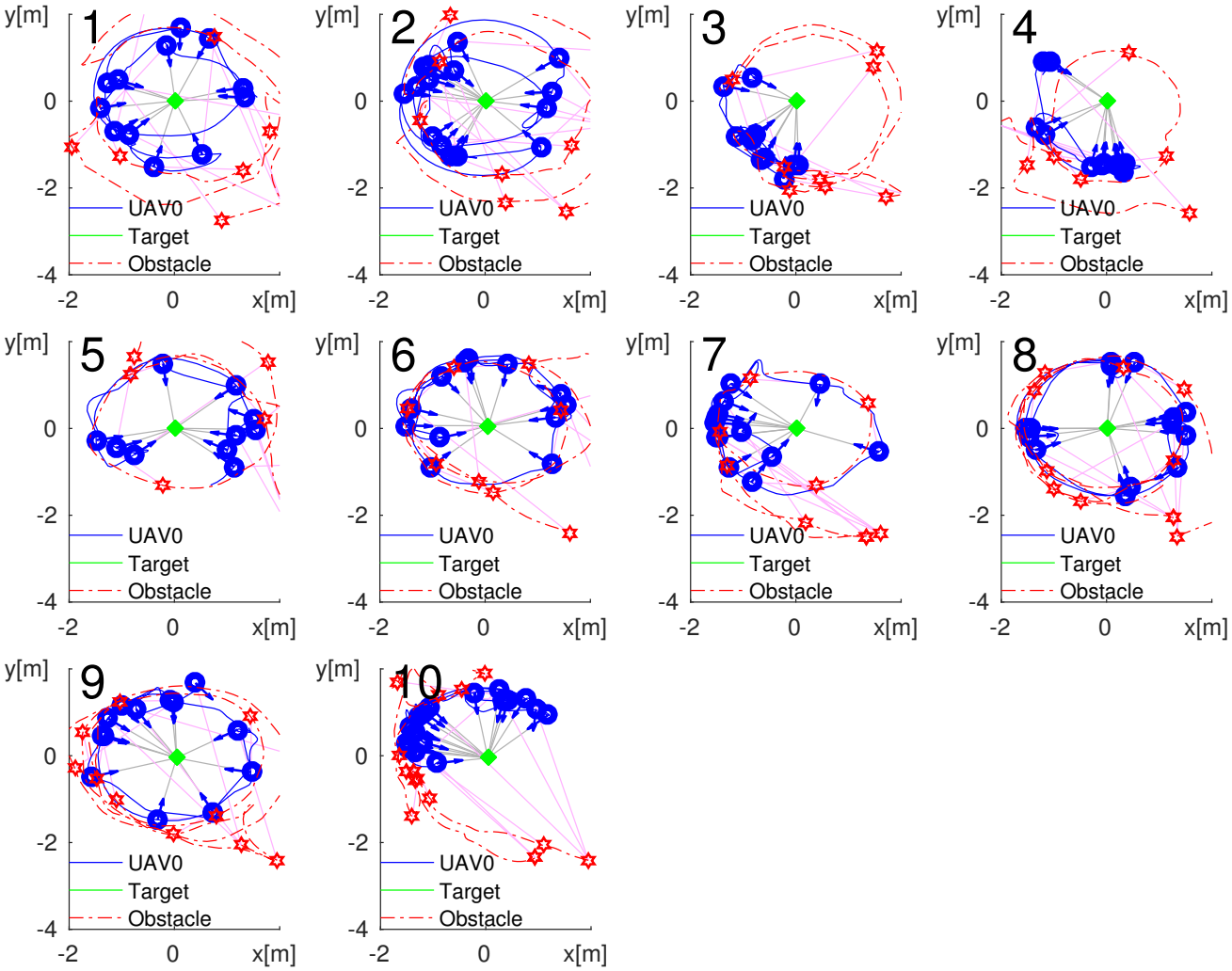


**Fig. 26** $UAV_0$ $xy$-trajectory with pose markers every $\Delta t = 4.2$s showing circular evasion maneuver pattern

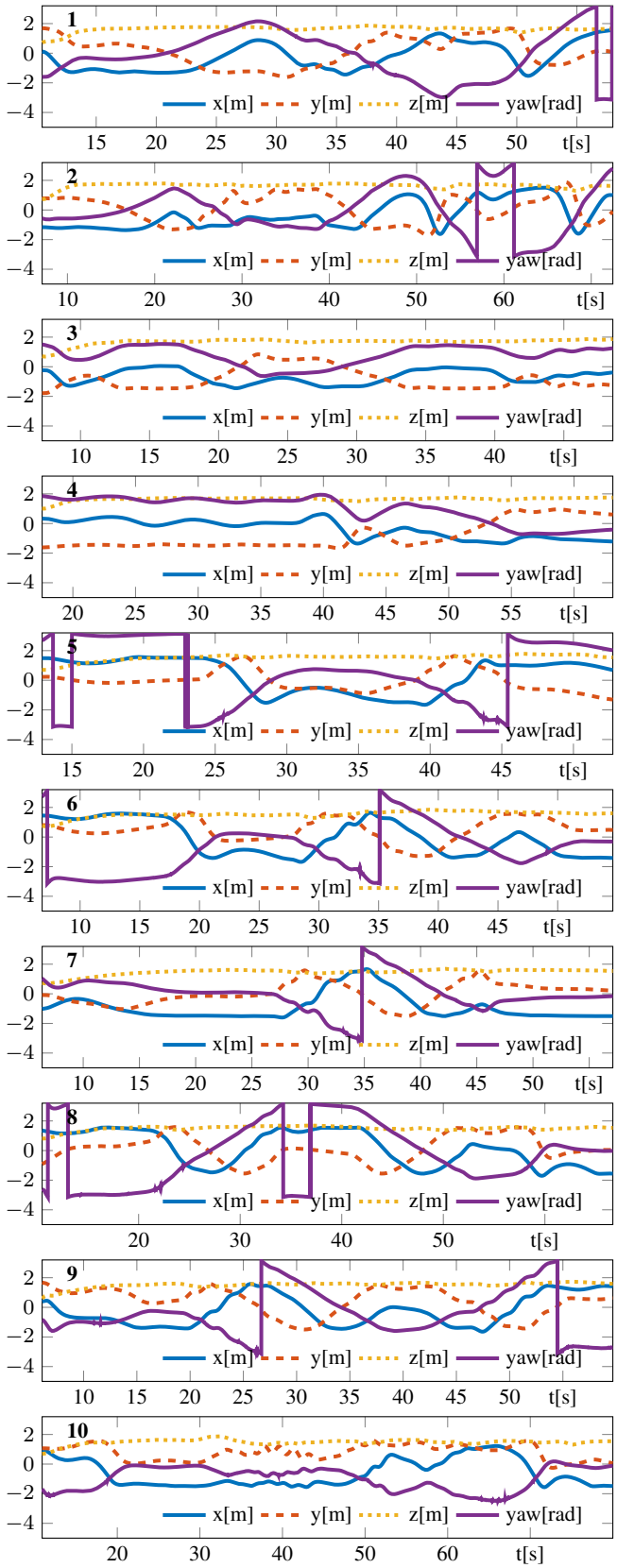


**Fig. 27** $UAV_0$ pose trajectory with steps in the $\Psi$-trajectory due to the limited yaw angle intervall

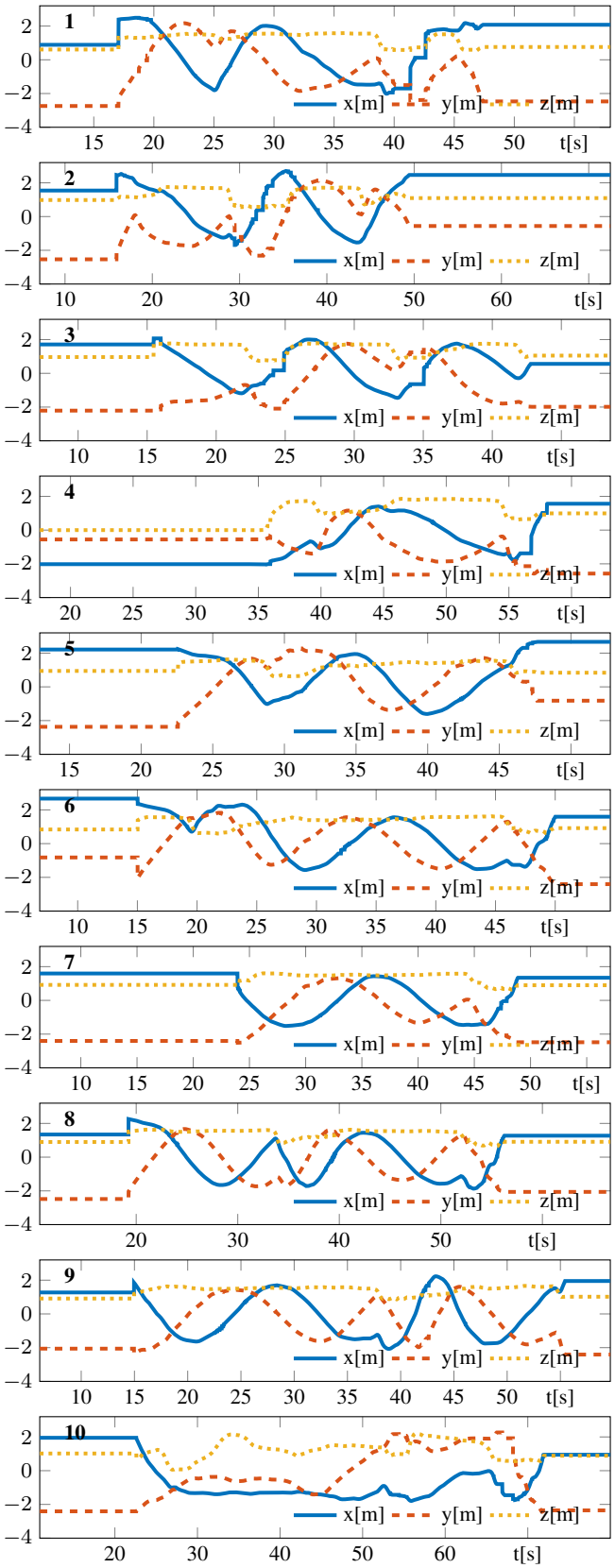


**Fig. 28** Manually introduced obstacle position showing circular pattern in order to follow $UAV_0$ on its evasion trajectory. Any steps are caused by entering the obstacle into the field of detection of the motion caption system.

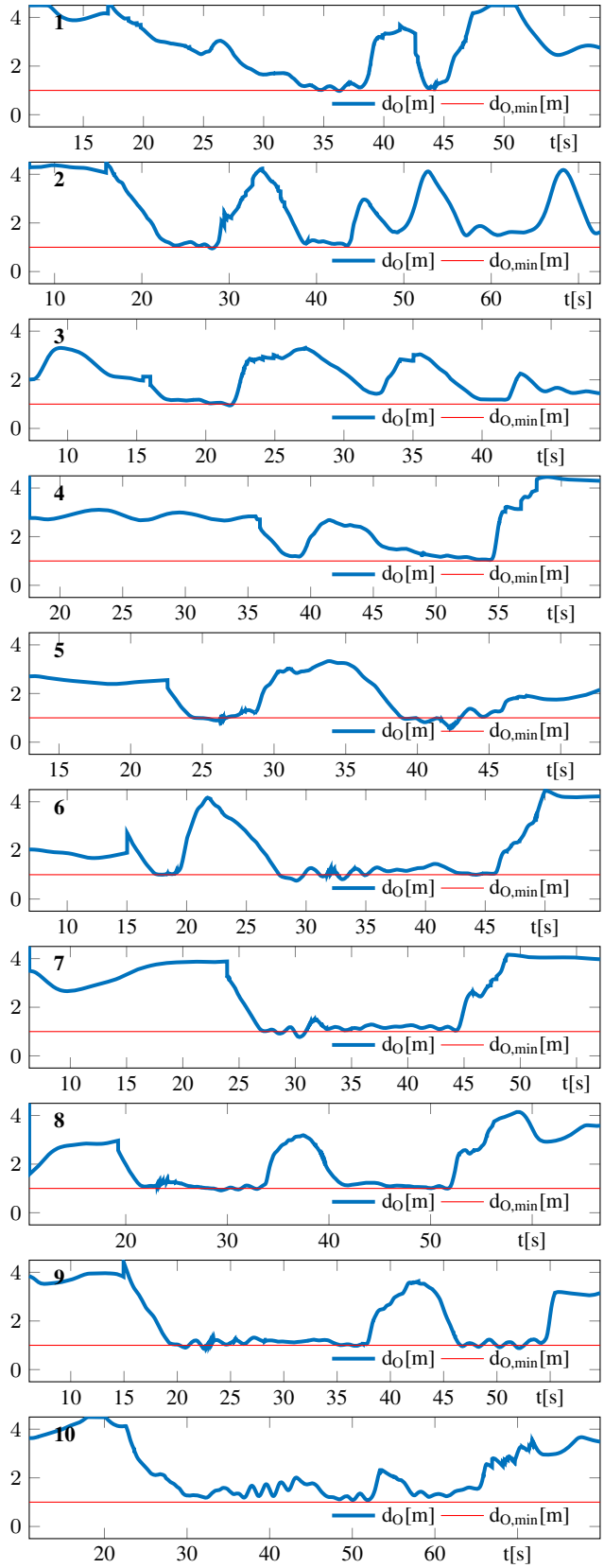


**Fig. 29** $UAV_0$ distance to obstacle $d_O$ stays above the defined minimum distance of $d_{O,min} = 1.0\text{m}$

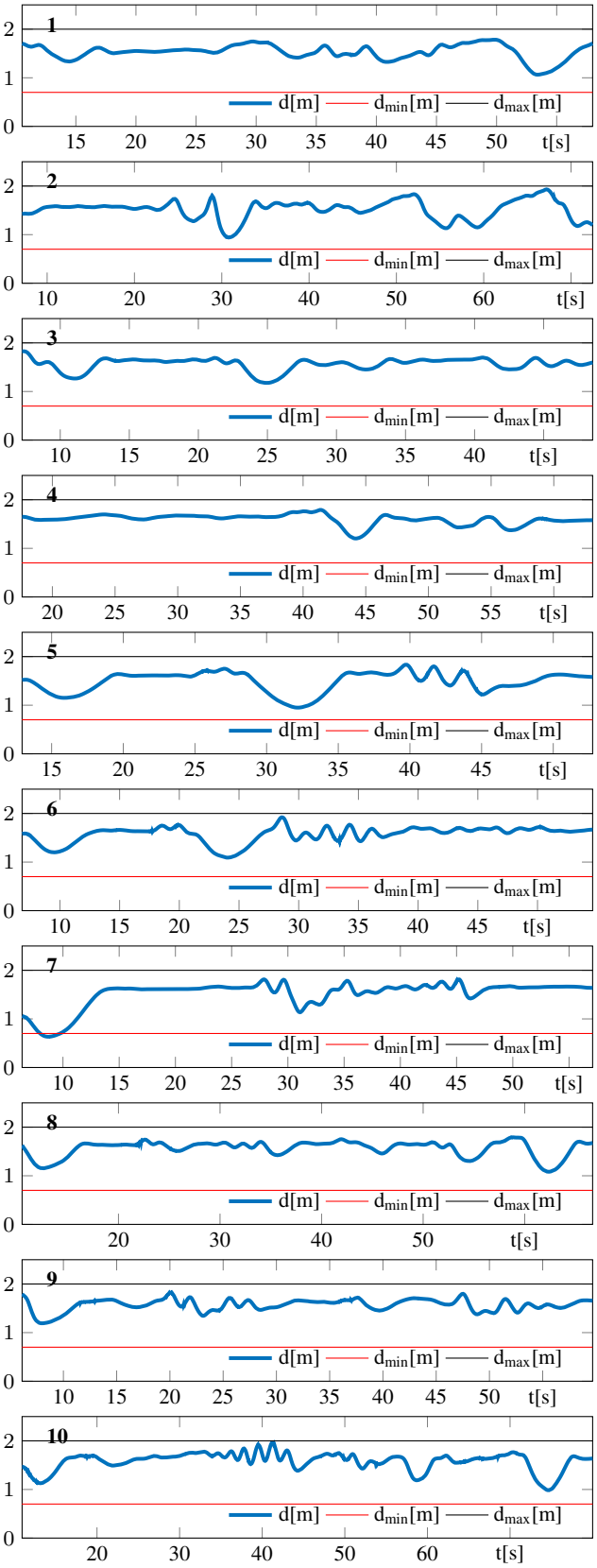


**Fig. 30** $UAV_0$ distance to target $d$ stays within the defined minimum distance of $d_{min} = 0.7\text{m}$ and maximum distance of $d_{max} = 2.0\text{m}$

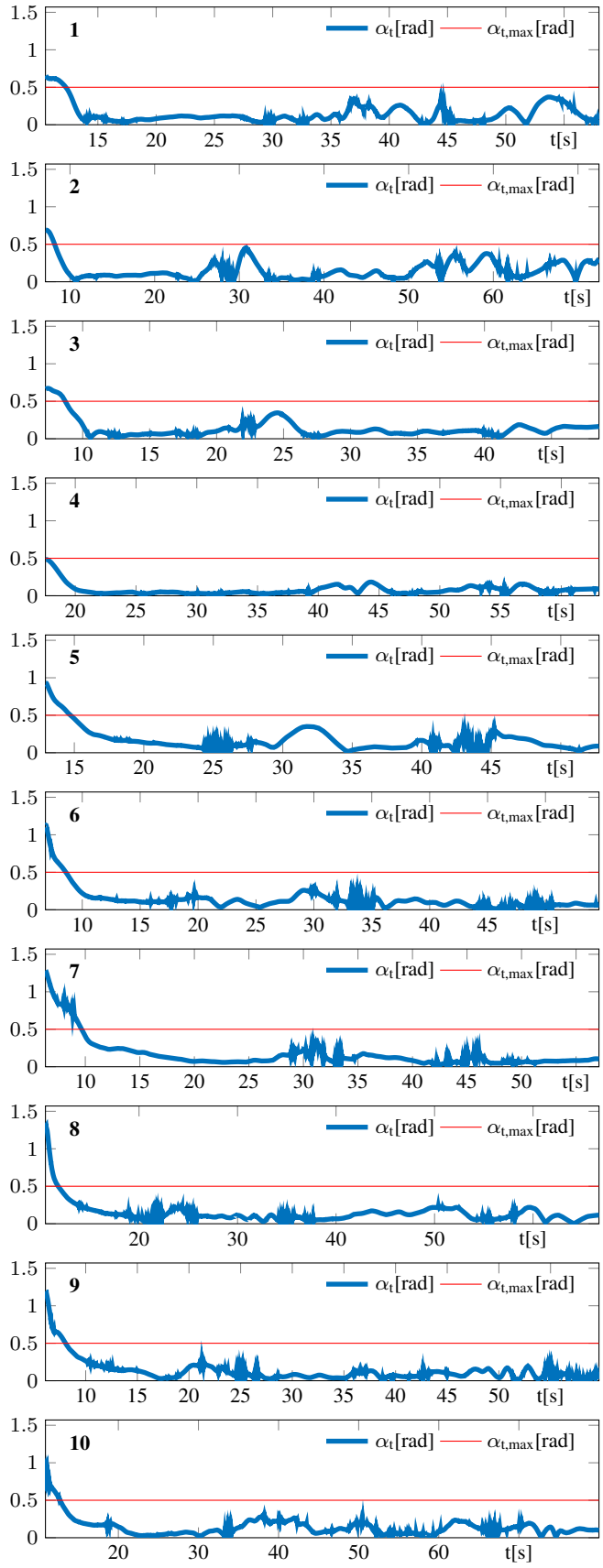


**Fig. 31** $UAV_0$ absolute tracking angle $\alpha_t$ stays within the defined maximum of $\alpha_{t,max} = 0.5$rad of the sensor cone constraint

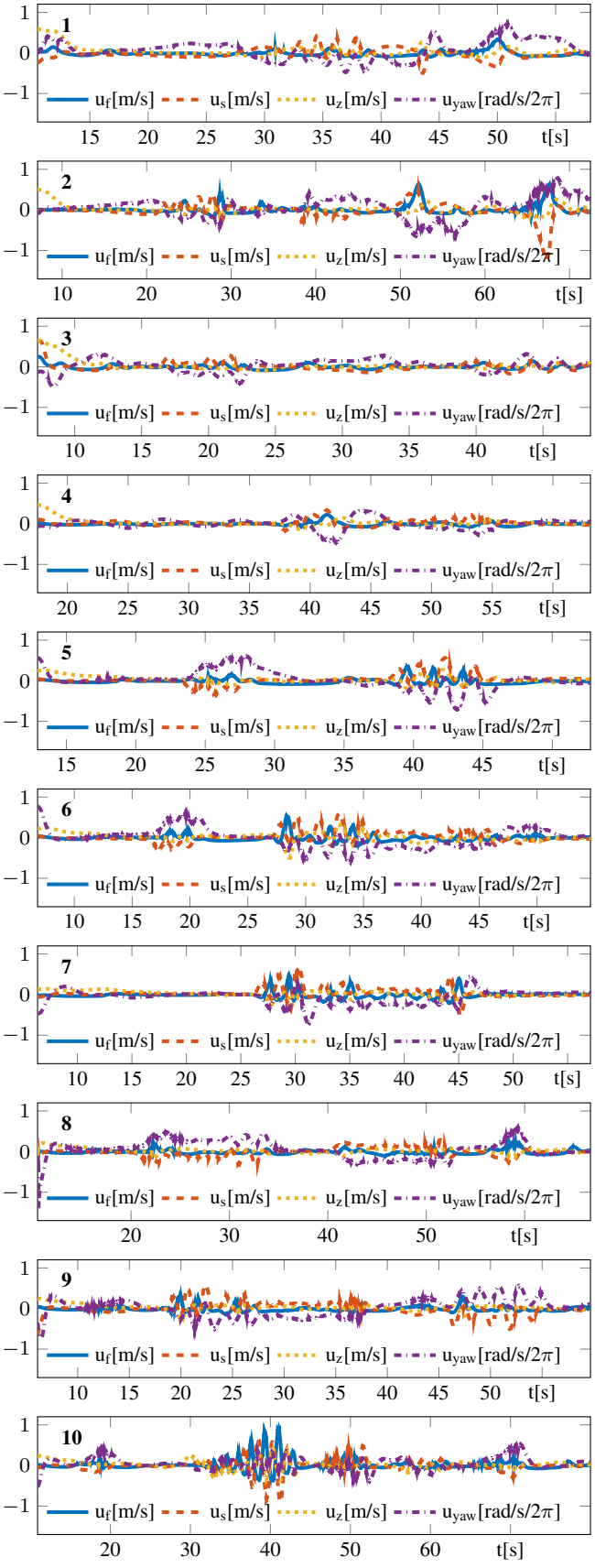


**Fig. 32** $UAV_0$ actuation

The influence of the target collision avoidance and cohesion constraints can be evaluated using the Euclidean distance of $UAV_0$ to the target, as shown in Fig. 30. Both constraints restrict the distance to $d_{min} = 0.7\text{m} \leq d \leq 2.0\text{m} = d_{max}$. The measured distance $d$ lies in the interval $0.941\text{m} \leq d \leq 1.966$ and therefore complies to the target distance constraints for all 10 experiments. The peak distance values are given in Table 7.3. The minimum tracking distance $\min(d) = 0.941\text{m}$ appears in run 2 and its maximum value $\max(d) = 1.966\text{m}$ in run 10.

Finally, the behavior of the developed sensor cone constraint (39) is validated. For this purpose the absolute tracking angle $\alpha_t$ (71)is shown in Fig. 31. The sensor constraint (39) is restricting the absolute tracking angle to $\alpha_t \leq \alpha_{t,max} = 0.5$. For the set of experiments, the initial conditions do typically not satisfy this constraint due to a low initial $UAV_0$ altitude $z$ and incompliant orientation $\psi$. This is directly shown by the controller counteraction as shown in Fig. 32. In the initial phase experiment 1-5 show a direct reaction in the altitude by $|u_z| >> 0$, while a significant adjustment in the $\psi$-axis by $|u_\psi| >> 0$ is dominating in experiment 5-10. The resulting convergence towards the constraint compliance $\alpha_t \to \alpha_{t,max}$ can be seen in all plots of Fig. 30 and is followed by a period of low action, as all constraints are satisfied and the obstacle is not considered yet. With the approaching obstacle $d_O \to d_{O,min}$ in Fig. 29, the control action in Fig. 32 is increased due to the evasion maneuver. As a result, also the tracking angle $\alpha_t$ in Fig. 31 is disturbed. To measure the constraint violation the maximal absolute tracking angle in the nominal state $\max(\alpha_{t,ns})$ is given in Table 7.3. For this purpose, $\max(\alpha_{t,ns})$ takes into consideration the absolute tracking angle peak values after the initial convergence phase. $\alpha_t \leq \alpha_{t,max} = 0.5\text{rad}$ holds after the initial phase for all the experiments. The measured maximum appears in run 2 with $\max(\alpha_{t,ns}) = 0.438\text{rad}$.

The experimental results validate the desired behavior of sensor based tracking also under the influence of disturbance introduced as obstacle. The senor field of view limitations are respected and collisions are avoided. Furthermore, the set of experiments shows the robust convergence towards a compliant state under differing initial conditions and constraint violations.

| run | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $\min(d_O)$[m] | 0.976 | 0.950 | 0.945 | 1.038 | **0.601** |
| $\min(d)$[m] | 1.064 | **0.941** | 1.176 | 1.203 | 0.950 |
| $\max(d)$[m] | 1.785 | 1.931 | 1.827 | 1.791 | 1.834 |
| $\max(\alpha_{t,ns})$[rad] | 0.371 | **0.438** | 0.345 | 0.183 | 0.350 |
| run | 6 | 7 | 8 | 9 | 10 |
| $\min(d_O)$[m] | 0.754 | 0.773 | 0.915 | 0.879 | 1.083 |
| $\min(d)$[m] | 1.091 | 0.633 | 1.082 | 1.191 | 0.984 |
| $\min(d)$[m] | 1.922 | 1.815 | 1.791 | 1.823 | **1.966** |
| $\max(\alpha_{t,ns})$[rad] | 0.260 | 0.307 | 0.220 | 0.225 | 0.271 |

**Table 1** Peak distance values for the set of 10 conducted experiments

## 8 Conclusion and Perspective

The present paper focuses on the presentation of a workflow for the generation of potential functions for sensor constrained $MPC$. The workflow is tested in simulation and a real-world implementation of a sensor constraint tracking scenario with quadrotors.

For this purpose a compact motion model for multi-rotor $UAV$s has been developed. To be able to control the robot's yaw angle orientation in $360°$, the problem of the discontinuously defined yaw angle $-\pi < \Psi \leq \pi$ has been addressed. The $MPC$ of the resulting direction vector model has been validated experimentally with an $AR.Drone\,2$ quadrotor.

Based on the developed system dynamic description, the workflow for a sensor constrained $MPC$ control has been presented. The $AR.Drone\,2$ quadrotor with attached sensor has served as platform for the sensor based tracking scenario. The first step in the workflow is to formulate the sensor perception space within a sensor coordinate frame by means of inequality constraints. For the use case, the cone shape of the sensor perception space has been described by a combination of two inequality constraints. As second step, these constraints have been transformed into a potential function with the help of unit steps. The idea is to introduce a repulsive behavior for a violation of the constraints which leads to a weakening of the constraints. This allows a small violation to maintaining feasibility of the $OCP$. The transformation into a weakened constraint with the help of unit steps helps to verify the cost functions without having to deal with the increased complexity of sigmoids. In addition, it allows a simple graphical verification of the potential function with visualization tools. To improve the solvability of the problem for gradient based solvers, the next step has been the introduction of a gradient in the undesired ares of the potential function which is pointing away from the target region. Subsequently, the previously introduced unit steps in the potential function are approximated with sigmoids which leads to a continuous gradient around the constraint borders. As a result the potential function becomes analytical and therefore solvable by standard real-time $MPC$ solvers.

In order to validate the derived potential function for the cone constraint, additional safety measures have been necessary. Therefore, a collision avoidance and a cohesion constraint have been developed, using the previously described workflow. In the validation scenario an $AR.Drone\,2.0$ is used to track another $AR.Drone\,2.0$ quadrotor which is following a circular trajectory. The tracking by means of the sensor constraint has been tested experimentally in simulation and a real-world implementation. To further discuss the influence of obstacles, utilized constraints and different initial conditions, a series of real-world collision avoidance scenarios has been presented. In the presented scenarios an

$AR.Drone\ 2.0$ is tracking a fixed target using the developed constraints for the cooperative control scenario while disturbance is introduced in form of an obstacle. The results show the desired collision evasion maneuver while maintaining sensor tracking for different initial conditions. The results have validated the developed multi-rotor prediction model as well as the sensor constraint potential function.

Future work will focus on the solution of the localization problem with cameras. A further development will be the analysis of the energy efficiency of the proposed sensor constraint, in contrast to a simple orientation tracking. Particularly interesting would be a statistical analysis of large numbers of real-world experiments. Another direction of future studies is to distribute the presented central $MPC$ and to explore the applicability of cloud computing. In addition, the application of the tracking scenario in heterogeneous systems e.g. ground robots and $UAV$s, will also be addressed in future work.

## References


1. N. Wingfield, "A field guide to civilian drones," https://www.nytimes.com/interactive/2015/technology/guide-to-civilian-drones.html, August 2016, accessed: 2017-04-18.
2. J. Dentler, S. Kannan, M. A. Olivares-Mendez, and H. Voos, "A real-time model predictive position control with collision avoidance for commercial low-cost quadrotors," in *2016 IEEE Multi-Conference on Systems and Control (MSC), Buenos Aires*, September 2016.
3. V. Gazi and K. M. Passino, *Swarm Coordination and Control Problems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 15–25. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-18041-5_2
4. Y. Zhang and H. Mehrjerdi, "A survey on multiple unmanned vehicles formation control and coordination: Normal and fault situations," in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, May 2013, pp. 1087–1096.
5. L. Consolini, F. Morbidi, D. Prattichizzo, and M. Tosques, "Leader-follower formation control of nonholonomic mobile robots with input constraints," *Automatica*, vol. 44, no. 5, pp. 1343–1349, 2008.
6. R. Cui, S. S. Ge, B. V. E. How, and Y. S. Choo, "Leader-follower formation control of underactuated auvs with leader position measurement," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 979–984.
7. S. Ge and Y. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous Robots*, vol. 13, no. 3, pp. 207–222, 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1020564024509
8. J. P. Desai, J. Ostrowski, and V. Kumar, "Controlling formations of multiple mobile robots," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 4, May 1998, pp. 2864–2869 vol.4.
9. A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, "A vision-based formation control framework," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 813–825, Oct 2002.
10. J. R. T. Lawton, R. W. Beard, and B. J. Young, "A decentralized approach to formation maneuvers," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 6, pp. 933–941, Dec 2003.
11. R. W. Beard, J. Lawton, and F. Y. Hadaegh, "A coordination architecture for spacecraft formation control," *IEEE Transactions on Control Systems Technology*, vol. 9, no. 6, pp. 777–790, Nov 2001.
12. H. Mehrjerdi, J. Ghommam, and M. Saad, "Nonlinear coordination control for a group of mobile robots using a virtual structure," *Mechatronics*, vol. 21, no. 7, pp. 1147 – 1155, 2011.
13. J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1465–1476, Sept 2004.
14. X. Cai and M. d. Queiroz, "Adaptive rigidity-based formation control for multirobotic vehicles with dynamics," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 1, pp. 389–396, Jan 2015.
15. W. Dong and J. A. Farrell, "Formation control of multiple underactuated surface vessels," *IET Control Theory Applications*, vol. 2, no. 12, pp. 1077–1085, December 2008.
16. T. P. Nascimento, A. G. S. Conceio, and A. P. Moreira, "Iterative weighted tuning for a nonlinear model predictive formation control," in *2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, May 2014, pp. 2–7.
17. T. P. Nascimento, A. P. Moreira, and A. G. S. Conceio, "Multi-robot nonlinear model predictive formation control: Moving target and target absence," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1502 – 1515, 2013.
18. J. Shin and H. J. Kim, "Nonlinear model predictive formation flight," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 39, no. 5, pp. 1116–1125, Sept 2009.
19. B. Alrifaee, M. G. Mamaghani, and D. Abel, "Centralized non-convex model predictive control for cooperative collision avoidance of networked vehicles," in *2014 IEEE International Symposium on Intelligent Control (ISIC)*, Oct 2014, pp. 1583–1588.
20. H. Fukushima, K. Kon, and F. Matsuno, "Model predictive formation control using branch-and-bound compatible with collision avoidance problems," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1308–1317, Oct 2013.
21. M. Turpin, N. Michael, and V. Kumar, "Decentralized formation control with variable shapes for aerial robots," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 23–30.
22. A. Bemporad and C. Rocchi, "Decentralized linear time-varying model predictive control of a formation of unmanned aerial vehicles," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, Dec 2011, pp. 7488–7493.
23. M. Diehl, H. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear mpc and moving horizon estimation," in *Nonlinear Model Predictive Control*, ser. Lecture Notes in Control and Information Sciences, L. Magni, D. Raimondo, and F. Allgöwer, Eds. Springer Berlin Heidelberg, 2009, vol. 384, pp. 391–417.
24. M. Geisert and N. Mansard, "Trajectory generation for quadrotor based systems using numerical optimal control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 2958–2964.
25. T. Ohtsuka, "A continuation/gmres method for fast computation of nonlinear receding horizon control," *Automatica*, vol. 40, no. 4, pp. 563 – 574, 2004.
26. ——, "symlab: Cgmres," http://www.symlab.sys.i.kyoto-u.ac.jp/ ohtsuka/code/index.htm, accessed: 2015-09-4.
27. M. Kang, T. Shen, and X. Jiao, "Continuation/gmres method based nonlinear model predictive control for {IC} engines," *{IFAC} Proceedings Volumes*, vol. 47, no. 3, pp. 5697 – 5702, 2014, 19th {IFAC} World Congress.
28. H. Seguchi and T. Ohtsuka, "Nonlinear receding horizon control of an underactuated hovercraft," vol. 13, no. 3-4. Wiley Online Library, 2003, pp. 381–398.

29. S. A. Sajadi-Alamdari, H. Voos, and M. Darouach, "Nonlinear model predictive extended eco-cruise control for battery electric vehicles," in *2016 24th Mediterranean Conference on Control and Automation (MED)*, June 2016, pp. 467–472.
30. Y. Shimizu, T. Ohtsuka, and M. Diehl, "A real-time algorithm for nonlinear receding horizon control using multiple shooting and continuation/krylov method," *International Journal of Robust and Nonlinear Control*, vol. 19, no. 8, pp. 919–936, 2009.
31. ——, "Nonlinear receding horizon control of an underactuated hovercraft with a multiple-shooting-based algorithm," in *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*, Oct 2006, pp. 603–607.
32. J. Dentler, S. Kannan, M. A. Olivares-Mendez, and H. Voos, "A modularization approach for nonlinear model predictive control of distributed fast systems," in *2016 24th Mediterranean Conference on Control and Automation (MED)*, June 2016, pp. 292–297.
33. ——, "Implementation and validation of an event-based real-time nonlinear model predictive control framework with ros interface for single and multi-robot systems," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, Aug 2017, pp. 1000–1006.
34. H. de Ruiter and B. Benhabib, "Visual-model-based, real-time 3d pose tracking for autonomous navigation: methodology and experiments," *Autonomous Robots*, vol. 25, no. 3, pp. 267–286, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10514-008-9094-7
35. F. Kendoul, K. Nonami, I. Fantoni, and R. Lozano, "An adaptive vision-based autopilot for mini flying machines guidance, navigation and control," *Autonomous Robots*, vol. 27, no. 3, p. 165, 2009. [Online]. Available: http://dx.doi.org/10.1007/s10514-009-9135-x
36. Ö. Erkent and H. Işıl Bozma, "Artificial potential functions based camera movements and visual behaviors in attentive robots," *Autonomous Robots*, vol. 32, no. 1, pp. 15–34, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10514-011-9240-5
37. J. Wang, J.-y. Liu, and H. Yi, "Formation control of unmanned surface vehicles with vision sensor constraints," *OCEANS 2015-MTS/IEEE Washington*, pp. 1–8, 2015.
38. S. Bhattacharya and S. Hutchinson, "Controllability and Properties of Optimal paths for a Differential Drive robot with field-of-view constraints," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, no. May, pp. 1624–1629, 2006.
39. S. Bhattacharya, R. Murrieta-Cid, and S. Hutchinson, "Optimal paths for landmark-based navigation by differential-drive vehicles with field-of-view constraints," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 47–59, 2007.
40. G. López-Nicolás, N. R. Gans, S. Bhattacharya, C. Sagüés, J. J. Guerrero, and S. Hutchinson, "Homography-based control scheme for mobile robots with nonholonomic and field-of-view constraints," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 40, no. 4, pp. 1115–1127, 2010.
41. W. Ding, M. R. Ganesh, R. N. Severinghaus, J. J. Corso, and D. Panagou, "Real-time model predictive control for keeping a quadrotor visible on the camera field-of-view of a ground robot," in *2016 American Control Conference (ACC)*, July 2016, pp. 2259–2264.
42. H. Seo, S. Kim, and H. J. Kim, "Aerial grasping of cylindrical object using visual servoing based on stochastic model predictive control," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 6362–6368.
43. G. B. Avanzini, A. M. Zanchettin, and P. Rocco, "Constraint-based Model Predictive Control for holonomic mobile manipulators," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-December, no. i, pp. 1473–1479, 2015.
44. M. Huang, H. Nakada, Butts, K. R., and I. V. Kolmanovsky, "Nonlinear Model Predictive Control of a Diesel Engine Air Path: A Comparison of Constraint Handling and Computational Strategies," *Proceedings - 5th IFAC Conference on Nonlinear Model Predictive Control - Seville, Spain*, 2015.
45. M. M. Lau and K. H. Lim, "Investigation of activation functions in deep belief network," in *2017 2nd International Conference on Control and Robotics Engineering (ICCRE)*, April 2017, pp. 201–206.
46. S. Bertrand, J. Marzat, H. Piet-Lahanier, A. Kahn, and Y. Rochefort, "MPC Strategies for Cooperative Guidance of Autonomous Vehicles," *AerospaceLab*, no. 8, pp. 1–18, 2014. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01103195
47. K. Graichen and B. Käpernick, "A real-time gradient method for nonlinear model predictive control," 2012.
48. P. Corke, *Robotics, Vision and Control*, 73rd ed., O. Khatib, Ed., 2013.
49. J. Dentler, "An event-based real-time nonlinear model predictive control framework," http://wiki.ros.org/denmpc, 2016, accessed: 2016-09-01.