Robocentric Model-based Visual Servoing for Quadrotor Flights

Yihang Li, *Student Member, IEEE,* Guozheng Lu, *Student Member, IEEE,* Dongjiao He, *Student Member, IEEE,* Fu Zhang, *Member, IEEE*

***Abstract*—This paper proposes a robocentric formulation for quadrotor visual servoing. This formulation presents the task-specific state dynamics of the quadrotor in its body reference frame. Compared to other visual servoing methods, our method allows tightly and integrated state estimation and control on the same robocentric model, and allows a faster system response in aggressive quadrotor flights. On the theory level, we prove the controllability and observability of the proposed robocentric model. Then, we design an on-manifold Kalman filter for the estimation and an on-manifold iterative model predictive controller (MPC) for motion planning and control. We verify our proposed formulation and controller in two crucial quadrotor flight tasks: hovering and dynamic obstacle avoidance. Experiment results show that the quadrotor is able to resist large external disturbances and recover its position and orientation from two reference visual features. Moreover, the quadrotor is able to avoid dynamic obstacles reliably at a relative speed up to 7.4 m/s, demonstrating the effectiveness of our visual servoing methods in agile quadrotor flights.**

***Index Terms*—Visual servoing, model predictive controller, obstacle avoidance, unmanned aerial vehicle.**

NOTATIONS

$\mathbf{x}$	State in aircraft's body frame.
$\hat{\mathbf{x}}$	Skew-symmetric matrix of $\mathbf{x}$.
$\mathbf{x}_m$	Direct measurement from IMU or camera.
$\mathbf{x}_h$	Stable state in aircraft's body frame.
$\mathbf{x}_{obs}$	State of dynamic obstacle represented in aircraft's body frame.

I. INTRODUCTION

UNMANNED aerial vehicles (UAVs) draw intensive attention from researchers and industry companies due to their great potential to various applications such as aerial photography, inspection, environment exploration and logistics. To fulfill these widespread applications, it is usually required to operate UAVs in close proximity of (or even in contact with) objects, such as flying through a cluttered environment with obstacles (e.g., narrow gaps), landing on a moving platform, perching on a target, and avoiding dynamic obstacles (e.g., birds). Achieving these agile flights reliably requires a low-latency processing pipeline consisting of the environment perception, ego-motion estimation and control. Such a low-latency pipeline allows a possibly faster response enabling the UAV to respond timely to any errors caused by external disturbances or internal model uncertainties.

Authors are with the Mechatronics and Robotic Systems (MaRS) Laboratory, Department of Mechanical Engineering, The University of Hong Kong, China. yhangli, gzlu, hdj65822, fuzhang@hku.hk

Fig. 1. Our quadrotor avoids a dynamic obstacle (a tennis ball), with relative speed up to 7.4 m/s. All perception, estimation, planning and control are performed by the onboard sensor (an Intel D455) and computer (a DJI Manifold 2 computer with Intel i7-8550U CPU). No prior knowledge of the size of the obstacle is required.

One way to achieve these mentioned agile flights is to separately estimate the UAV's ego-motion and the target (e.g., a landing pad) or the surrounding environment. This typically requires a full simultaneous localization and mapping (SLAM) or visual(-inertial) odometry in GPS denied environments. Then, a trajectory planning is performed to find a feasible and collision-free trajectory from the UAV's current state to the target one. This trajectory is then tracked by an underlying tracking controller. Despite its separation of different modules and generality to different tasks, this full pipeline typically requires significant computation resources and leads to considerably high latency.

Visual servoing techniques provide an alternative solution for UAV navigation and control, which directly use image data in the servo loop to control the motion of a robot [1]. Compared to the above mentioned full pipeline, visual servoing establishes a direct estimation between the robots with the target. This more tightly-coupled and compact pipeline promises lower latency from perception to control, enabling the UAV to execute mission with high accuracy and aggressiveness.

Generally, visual servoing is classified into two categories: position-based visual servoing (PBVS) and image-based visual servoing (IBVS) [1], [2]. PBVS methods first estimate the robot's position (and other states) in the world frame and then perform the control based on the estimated states. The robot's position could be estimated from the projection of known structures, such as collinear points, spheres, cylinders and multiple heterogeneous features [3], or a more general visual(-inertial) odometry [4] by robustly tracking salient

visual features in the environment. In the latter case, the PBVS reduces to the common pipeline which requires a complete state estimation in the world frame thus suffers from similar drawbacks of long processing pipeline and increased computation load and latency.

In contrast to PBVS methods, IBVS methods do not require to estimate explicitly the robot's position (or other states) in a world frame, but directly take control to make a number of visual features reach their specified values. To achieve this, the visual features are first defined according to a specific task. Then the time derivatives of the visual features are obtained, which are related to the robot's full velocity. Based on this differential equation relating the time derivative of the visual features and the robot's velocity, a control law can be easily designed to drive the visual features to their desired values by viewing the robot's velocity as the virtual control.

By utilizing the visual features for control, the IBVS methods promise a lower latency and have enabled many interesting UAV applications recently. In [5][6], UAVs were able to hover at a desired position utilizing the IBVS method, where the visual features were defined as the image moments on a virtual image plane. With the defined visual features, the authors designed a cascaded controller and further proved its stability. In [7], the authors demonstrated a quadrotor flying through multiple gaps using the IBVS method. The reciprocal of distance between every two projected visual points on a unit sphere was used as the visual features. The specially designed visual features were invariant to robot rotation thus the rotation dynamics of the quadrotor was decoupled. Path planning and control were designed and implemented which enable the quadrotor to fly through narrow gaps. IBVS was also implemented on a quadrotor to land on a moving platform in [8]. The centroid of landmarks' projection on a sphere in body frame was selected as the visual features. Different from other IBVS works which view static reference target as landmarks, the platform for landing is moving. They additionally used optical flow technique to estimate the relative velocity between the quadrotor and the moving platform. Thomas *et al.* [9] considered a perching task. The coordinates of a cylinder projection on a virtual plane were selected as the visual features. A perching trajectory was generated and tracked then the quadrotor was able to perch on the target cylinder. The selection of visual features in this work also decoupled the angular dynamics from the visual feature dynamics thus a cascaded controller was designed to separately control position and orientation. IBVS methods also found some applications in the task of UAV dynamic obstacle avoidance, a very challenging task that has drawn much recent research interest [10]–[14]. In [15], authors designed a nonlinear MPC (NMPC)-based IBVS method to enable a fixed-wing UAV to avoid a dynamic obstacle in simulation. Mcfadyen *et al.* [16] applied the IBVS method to a quadrotor UAV for dynamic obstacle avoidance but only demonstrated the system with a virtual, simulated dynamic obstacle when the UAV was tracking a conical spiral trajectory.

The IBVS methods mentioned above typically differ in the selection of visual features, but the controllers all use the robot's velocity [5], [16] or acceleration [6], [7], [9], [15], [17] as the virtual control, requiring an additional cascaded attitude controller to track the computed velocity or acceleration commands. In this cascaded control structure, the visual servoing controller essentially neglects the underlying dynamics caused by the inner-loop attitude controller and hence limits the system's attainable response rate.

Regardless of the navigation and control framework (visual servoing or non-visual servoing, PBVS or IBVS), a key problem is how to design an appropriate controller within the chosen framework. In recent years, control theory has been widely applied in real-world nonlinear systems, such as H-infinity control [18], [19], fuzzy control [20], [21], sliding mode control [22], model predictive control (MPC) [23], [24] and learning-based control [25], [26]. Among these control methods, MPC transforms the control problem to an optimization problem considering both input and state constraints. These properties have made MPC very flexible to incorporate actual constraints (e.g., actuation limit) and been popularly used in real-world robotic systems. In [27], [28], MPC was used to achieve tracking control with a given trajectory. In [11], [23], collision-free constraint was considered within MPC and in [6] a tightened state constraint was included for external disturbance rejection. Authors in [24] proposed an iterative method, sequential quadratic programming (SQP), as the optimization solver for MPC applied on a quadrotor UAV. MPC has also been used in previous visual servoing methods. In the framework of visual servoing, [29] used MPC to handle input and visibility constraints and [30] applied the MPC method on underwater vehicles for setpoint tracking. In [31] and [6], MPC was utilized as visual servoing controller on quadrotors and they focus on visibility improvement and external disturbance rejection respectively. While these works designed the MPC in normal Euclidean space, our previous work [32] considered how to incorporate the manifold constraints into a trajectory tracking MPC controller since real-world robotic systems usually evolve on manifolds.

In this work, we propose a robocentric approach for the IBVS method. The key idea is to establish a full dynamic model (as opposed to the robot's velocity only) of the visual features in the robot's body reference frame (as opposed to the world frame where the robot's velocity is expressed). The consideration of the full dynamic model allows a faster response of the UAV, while the robocentric model agrees well with the visual features which are naturally expressed in the body reference frame (e.g., the image frame). More specifically, our contributions are as follows:

1) We propose a robocentric model for quadrotor visual servoing control and analyze its controllability and observability. Compared with existing IBVS methods [5]–[9], [15], [16], which use simplified models actuated by the UAV's velocity or acceleration, our model is a full dynamic model actuated directly by the UAV's angular velocity, allowing a possibly faster system response.
2) Based on the robocentric model, we implement an integrated estimation and visual servoing control framework. The estimation is a lightweight on-manifold state estimator that utilizes measurements of only onboard depth-camera and IMU to estimate the full robocentric state. For the visual servoing controller, we propose an on-manifold iterative Model Predictive Controller (iMPC).

Compared with previous on-manifold MPC [32] which always requires a feasible trajectory planned beforehand and [24] that parameterized the attitude using redundant parameters (i.e., quaternion), our proposed iMPC is able to perform trajectory planning and tracking control jointly while admitting manifold constraints.

3) The proposed formulation and method are verified by experiments on a quadrotor UAV. Besides hovering tasks, we apply our method to more challenging dynamic obstacle avoidance task. To our best knowledge, it is the first time that a visual servoing controller is successfully demonstrated for such task with real-world UAVs and environments, which differs from the simulated environments or virtual obstacles demonstrated in existing visual servoing methods [15], [16].

This paper is organized as follows: Section II derives the robocentric model in a given task and analyzes its controllability and observability. Section III presents the integrated estimation and control methods based on the robocentric model. Section IV presents simulation and experiment results of our methods in two quadrotor tasks: hovering and dynamic obstacle avoidance. Section V draws conclusions and discusses future work.

II. Robocentric Modeling

In this section, states of a quadrotor will be represented in body frame (i.e., the front-right-down frame). We consider two tasks, hovering and dynamic obstacle avoidance. For each task, the corresponding visual targets are selected and system equations are derived respectively. When deriving the system state equation, we assume the control inputs are the collective thrust acceleration a_T and the body angular velocity $\boldsymbol{\omega}$, which can be tracked by a cascaded inner-loop angular velocity controller. Moreover, we assume a RGB-D sensor configuration on the quadrotor UAV, so that the full coordinates of the visual targets in the body frame are directly measurable. Although we consider these two tasks for presentation clarity, the presented robocentric formulation can be extended to other visual servoing tasks with similar derivations.

A. Hover

The hovering of a quadrotor can be achieved by choosing a static visual target and maintaining its RGB-D measurement (coordinates in the body frame) at a given value. Assume the measurement of the visual target is $\mathbf{p}$, which is represented in the quadrotor body frame. To maintain this measurement at a given position $\mathbf{p}_h$, we derive, based on the standard rigid-body motion [33], its state equation with respect to the input collective thrust acceleration a_T and body angular velocity $\boldsymbol{\omega}$, as follows:

$$\dot{\mathbf{p}} = -\hat{\boldsymbol{\omega}}\mathbf{p} - \mathbf{v} \quad \dot{\mathbf{v}} = -\hat{\boldsymbol{\omega}}\mathbf{v} + \mathbf{a} + \mathbf{g} \quad \dot{\mathbf{g}} = -\hat{\boldsymbol{\omega}}\mathbf{g} \tag{1}$$

where $\mathbf{v}$ denotes the velocity, $\mathbf{a} = -a_T\mathbf{e}_3$ denotes the acceleration along the UAV belly with $\mathbf{e}_3 = [0,0,1]^T$, and $\mathbf{g}$ denotes the gravity, all in the body frame, $\hat{\boldsymbol{\omega}}$ is skew-symmetric matrix of $\boldsymbol{\omega}$. Based on the state representation in (1), the hovering task is achieved by stabilizing the state around the equilibrium point at $(\mathbf{p}_h, \mathbf{0}, g\mathbf{e}_3)$, where $g \approx 9.8$ m/s^2 is the gravity.

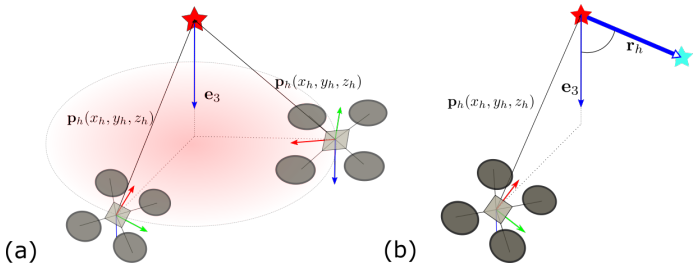


Fig. 2. (a) The quadrotor may hover at any position on a horizontal circle when the only one target point is controlled. (b) The quadrotor hovers at a determined position with control of one visual target and one reference direction (e.g., made up by two visual targets).

The visual servoing formulation in (1) cannot determine a unique pose of the quadrotor in the world frame. As shown in Fig. 2(a), the quadrotor would freely move along a horizontal circle of radius $(x_h^2 + y_h^2)^{\frac{1}{2}}$ around the point $\mathbf{p}_h$ without changing the measurement of the visual targets at all.

In order to fully control the quadrotor orientation during a hovering, we consider another visual target $\mathbf{p}_j$ in addition to the first one denoted as $\mathbf{p}_i$. A unit direction vector is made up by the two visual targets $\mathbf{r} = \frac{\mathbf{p}_j - \mathbf{p}_i}{\|\mathbf{p}_j - \mathbf{p}_i\|} \in \mathbb{S}^2$. Similar to gravity, this direction provides extra orientation information of the quadrotor, and maintaining the observed direction $\mathbf{r}$ at a desired value $\mathbf{r}_h$ would uniquely determine the quadrotor orientation in the world frame. To achieve this control objective, we need to augment the state dynamics in (1) further by that of the direction $\mathbf{r}$, as follows:

$$\begin{aligned} \dot{\mathbf{p}} = -\hat{\boldsymbol{\omega}}\mathbf{p} - \mathbf{v} \quad \dot{\mathbf{v}} = -\hat{\boldsymbol{\omega}}\mathbf{v} + \mathbf{a} + \mathbf{g} \\ \dot{\mathbf{g}} = -\hat{\boldsymbol{\omega}}\mathbf{g} \quad \dot{\mathbf{r}} = -\hat{\boldsymbol{\omega}}\mathbf{r} \end{aligned} \tag{2}$$

One thing to notice is that the choice of $\mathbf{r}_h \in \mathbb{S}^2$ is not completely free because it has to respect the simple geometric constraint that, the angle between $\mathbf{r}_h$ and the gravity at the desired state (i.e., hovering), $\mathbf{g}_h = g\mathbf{e}_3$, has to be equal to the actual one. That is,

$$\mathbf{r}_h^T \cdot \mathbf{g}_h = c \tag{3}$$

where c is a constant determined by the actual angle between the reference direction and the gravity in the space. It is within the ranges $(-g, g)$ and its value is estimated from the first few measurements of the reference direction $\mathbf{r}$ and the estimation of the gravity vector $\mathbf{g}$. Violating the constraint in (3) would lead the desired direction $\mathbf{r}_h$ unattainable at hovering.

On the other hand, the choice of $\mathbf{p}_h$ is relatively free as long as it is within the sensor FoV. As a consequence, $\mathbf{p}_h$ contributes three degrees of freedom, and $\mathbf{r}_h$ contributes just one due to the constraint in (3). The total four degrees of freedom are equal to the controllable degrees of freedom of a quadrotor. As introduced in [1], to control the 6 DOF of the system, it is necessary to consider at least three visual targets in visual servoing methods. However, quadrotor is an under-actuated system and its 6 DOF cannot be simultaneously controlled. In our method, the under-actuated dynamics of the quadrotor is considered thus using only two visual targets is

able to control the system. When the desired coordinates of the visual target $\mathbf{p}_h$ and the reference direction $\mathbf{r}_h$ are determined in the body frame, the quadrotor has a unique position and orientation as shown in Fig. 2(b).

Denote $\mathbf{x} = (\mathbf{p}, \mathbf{v}, \mathbf{g}, \mathbf{r})$ the state and $\mathbf{u} = (\boldsymbol{\omega}, a_T)$ the control input of the system in (2), we have the following theorem.

Theorem 1. *The pair* $(\mathbf{x}_h, \mathbf{u}_h)$*,* $\mathbf{x}_h = (\mathbf{p}_h, \mathbf{0}, g\mathbf{e}_3, \mathbf{r}_h)$ *with* $\mathbf{r}_h$ *satisfying (3) and* $\mathbf{u}_h = (\mathbf{0}, g)$*, represents an equilibrium point of the system (2). Moreover, the system (2) around the point* $(\mathbf{x}_h, \mathbf{u}_h)$ *is locally controllable.*

Proof. The controllability proof can be referred at Sec. I in our supplementary material [34]. □

B. Obstacle Avoidance

To make sure the quadrotor to hover safely although with the existence of some dynamic obstacles which may collide with the quadrotor, the system model is extended by including an obstacle's dynamics. In our circumstance, only one obstacle is considered and its relative position to quadrotor is measured and denoted in the quadrotor body frame as $\mathbf{p}_{obs}$. Its velocity is denoted as $\mathbf{v}_{obs}$. The dynamics of the obstacle is established by assuming only gravity affecting its motion and it can be written in quadrotor robocentric formulation as

$$\dot{\mathbf{p}}_{obs} = -\hat{\boldsymbol{\omega}}\mathbf{p}_{obs} - (\mathbf{v} - \mathbf{v}_{obs}) \quad \dot{\mathbf{v}}_{obs} = -\hat{\boldsymbol{\omega}}\mathbf{v}_{obs} + \mathbf{g} \tag{4}$$

In the above obstacle dynamics, it can be seen that $\mathbf{v}$ and $\boldsymbol{\omega}$ are included in right hand side. States of the obstacle (i.e. $\mathbf{p}_{obs}$ and $\mathbf{v}_{obs}$) are directly related to states of the quadrotor thus obstacle avoidance can be considered as a control problem. Combining (2) with above obstacle dynamics leads to

$$\begin{aligned} &\dot{\mathbf{p}} = -\hat{\boldsymbol{\omega}}\mathbf{p} - \mathbf{v} \quad \dot{\mathbf{v}} = -\hat{\boldsymbol{\omega}}\mathbf{v} + \mathbf{a} + \mathbf{g} \quad \dot{\mathbf{g}} = -\hat{\boldsymbol{\omega}}\mathbf{g} \quad \dot{\mathbf{r}} = -\hat{\boldsymbol{\omega}}\mathbf{r} \\ &\dot{\mathbf{p}}_{obs} = -\hat{\boldsymbol{\omega}}\mathbf{p}_{obs} - (\mathbf{v} - \mathbf{v}_{obs}) \quad \dot{\mathbf{v}}_{obs} = -\hat{\boldsymbol{\omega}}\mathbf{v}_{obs} + \mathbf{g} \end{aligned} \tag{5}$$

The goal of this task is to avoid a dynamic obstacle from a hover state and recover to hover thereafter. In this problem, $\mathbf{p}_{obs}$ is not required to reach some desired value, but to stay above a safe clearance. Therefore the controllability of the above system reduces to the controllability of the hovering state, which is already proved controllable in Section II-A.

III. Integrated Estimation and Iterative Model Predictive Control

When compared to existing visual servoing methods, one benefit of the robocentric formulation (e.g., (2) for hovering and (5) for obstacle avoidance) is that, it allows integrated estimation and control based on the same task-specific state equation, without using an additional visual-inertial odometry (VIO). In this section, state estimation of our system is introduced firstly. Then we propose an on-manifold iterative Model Predictive Control (iMPC). We will introduce its solving procedures and then specify the control goals of the robocentric tasks.

A. State Estimation

All states in our method for quadrotor control are estimated by a stereo RGB-D camera mounted on the quadrotor and a 6-axis IMU. Positions of visual targets are measured in the camera frame, which is transformed to the body frame $\mathbf{p}$ from the known extrinsic parameters. The IMU measurements contain the angular velocity $\boldsymbol{\omega}_m$ and acceleration $\mathbf{a}_m$. Augmenting the state equation in (2) with the IMU model [35] leads to the complete state equation for the hovering task:

$$\begin{aligned} &\dot{\mathbf{p}} = \hat{\mathbf{p}}\,(\boldsymbol{\omega}_m - \mathbf{b}_g - \mathbf{n}_g) - \mathbf{v} \quad \dot{\mathbf{b}}_g = \mathbf{n}_{bg} \quad \dot{\mathbf{b}}_a = \mathbf{n}_{ba} \\ &\dot{\mathbf{v}} = \hat{\mathbf{v}}\,(\boldsymbol{\omega}_m - \mathbf{b}_g - \mathbf{n}_g) + \mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a + \mathbf{g} \\ &\dot{\mathbf{g}} = \hat{\mathbf{g}}\,(\boldsymbol{\omega}_m - \mathbf{b}_g - \mathbf{n}_g) \quad \dot{\mathbf{r}} = \hat{\mathbf{r}}\,(\boldsymbol{\omega}_m - \mathbf{b}_g - \mathbf{n}_g) \end{aligned} \tag{6}$$

where $\hat{\mathbf{p}}, \hat{\mathbf{v}}, \hat{\mathbf{g}}, \hat{\mathbf{r}}$ are skew-symmetric matrix of $\mathbf{p}, \mathbf{v}, \mathbf{g}, \mathbf{r}$ respectively and $\mathbf{b}_g$ and $\mathbf{b}_a$ are the gyro and accelerometer bias, respectively, which are typically modelled as random walks [35]. $\mathbf{n}_g$, $\mathbf{n}_a$, $\mathbf{n}_{bg}$, $\mathbf{n}_{ba}$ are all noises. The measurements for the hovering task contain the measurement of the visual target $\mathbf{p}_m$ and the reference direction $\mathbf{r}_m$:

$$\mathbf{p}_m = \mathbf{p} + \mathbf{n}_{\mathbf{p}} \quad \mathbf{r}_m = \mathbf{r} + \mathbf{n}_{\mathbf{r}} \tag{7}$$

where $\mathbf{n}_{\mathbf{p}}$, $\mathbf{n}_{\mathbf{r}}$ are measurement noises.

Theorem 2. *The system with states (6) and measurements (7) is locally weakly observable for* $\mathbf{p} \neq \mathbf{0}$.

Proof. The observability proof is based on the on-manifold observability analysis in [36] and the detailed proof can be referred at Sec. II of our supplementary material [34]. □

For the task of dynamic obstacle avoidance, the full state equations can be obtained similarly by augmenting the IMU states. The resultant system is similar to (6) but contains two more states, $\mathbf{p}_{obs}$ and $\mathbf{v}_{obs}$. These two more states can be trivially estimated from the additional measurement of $\mathbf{p}_{obs}$, thus does not alter the observability of (6). For example, $\mathbf{p}_{obs}$ can directly use its measured value and $\mathbf{v}_{obs}$ can be obtained by differentiating $\mathbf{p}_{obs}$. To sum up, the system remains observable.

With the proved observability of the two systems, their states (i.e., $(\mathbf{p}, \mathbf{v}, \mathbf{g}, \mathbf{r}, \mathbf{b}_g, \mathbf{b}_a)$ for hovering and $(\mathbf{p}, \mathbf{v}, \mathbf{p}_{obs}, \mathbf{v}_{obs}, \mathbf{g}, \mathbf{r}, \mathbf{b}_g, \mathbf{b}_a)$ for dynamic obstacle avoidance) are estimated by an iterated Kalman filter that operates on the respective state manifolds. We directly use the on-manifold extended Kalman filter implementation in our previous work [37] by simply specifying the manifold of each system and casting the system into a canonical representation. Details of the Kalman filter implementation can be referred to [37].

B. Canonical representation

With the state estimation in the previous section, we have the knowledge of the full state vector, including the IMU bias terms. In this section, we design a model predictive controller (MPC) for each task based on the state feedback. With the knowledge of the bias terms, we estimate the actual angular velocity by $\boldsymbol{\omega} = \boldsymbol{\omega}_m - \mathbf{b}_g$, the collective thrust acceleration by $a_T = \|\mathbf{a}_m - \mathbf{b}_a\|$ and use them to track any commands computed by the MPC. Therefore, the MPC is designed on state

space model without the bias terms (i.e., (2) for hovering and (5) for obstacle avoidance) and has the control input $\mathbf{u} = (\boldsymbol{\omega}, a_T)$.

To design a hovering controller that runs at discrete time, we discretize the system (2) using the Euler's method as below:

$$\underbrace{\begin{bmatrix}\mathbf{p}_{k+1}\\ \mathbf{v}_{k+1}\\ \mathbf{g}_{k+1}\\ \mathbf{r}_{k+1}\end{bmatrix}}_{\mathbf{x}_{k+1}} = \underbrace{\begin{bmatrix}\mathbf{p}_k + \Delta t(-\hat{\boldsymbol{\omega}}_k\ \mathbf{p}_k - \mathbf{v}_k)\\ \mathbf{v}_k + \Delta t(-\hat{\boldsymbol{\omega}}_k\ \mathbf{v}_k + \mathbf{a}_k + \mathbf{g}_k)\\ \mathbf{Exp}(-\Delta t\boldsymbol{\omega}_k)\ \mathbf{g}_k\\ \mathbf{Exp}(-\Delta t\boldsymbol{\omega}_k)\ \mathbf{r}_k\end{bmatrix}}_{\mathbf{x}_k \oplus (\Delta t\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k))} \tag{8}$$

where $\mathbf{Exp}(\boldsymbol{\theta}) = \mathbf{I} + \frac{\hat{\boldsymbol{\theta}}}{\|\boldsymbol{\theta}\|}\sin\|\boldsymbol{\theta}\| + \left(\frac{\hat{\boldsymbol{\theta}}}{\|\boldsymbol{\theta}\|}\right)^2(1-\cos\|\boldsymbol{\theta}\|)$ is the exponential function on $SO(3)$ transforming an axis-angle vector to a rotation matrix [38] and Δt represents the sampling time, $\mathbf{x}_k$ denotes the entire system state and $\mathbf{u}_k$ denotes the control input, both at the time step k. As can be seen, the discrete system naturally evolves on the state manifold $\mathcal{M} = \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{S}^2 \times \mathbb{S}^2$ with dimension $dim(\mathcal{M}) = 10$.

The discrete system in (8) is written into a more compact form of $\mathbf{x}_{k+1} = \mathbf{x}_k \oplus (\Delta t\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k))$, where the operator $\oplus$ encodes compactly the "addition" of a state at its present time and the state perturbation caused by the input for one sampling period, such that the state at the next step still remains on the state manifold $\mathcal{M}$. Specially, for system evolving in $\mathbb{R}^n$, $\oplus$ is the usual vector addition. Finally, $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \in \mathbb{R}^r$ represents the state perturbation caused by the input and is defined as:

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix}-\hat{\boldsymbol{\omega}}\mathbf{p} - \mathbf{v}\\ -\hat{\boldsymbol{\omega}}\mathbf{v} - a_T\mathbf{e}_3 + \mathbf{g}\\ -\boldsymbol{\omega}\\ -\boldsymbol{\omega}\end{bmatrix} \in \mathbb{R}^{12} \tag{9}$$

For task of obstacle avoidance, the discrete state equation can be derived similarly:

$$\underbrace{\begin{bmatrix}\mathbf{p}_{k+1}\\ \mathbf{v}_{k+1}\\ \mathbf{p}_{obs,k+1}\\ \mathbf{v}_{obs,k+1}\\ \mathbf{g}_{k+1}\\ \mathbf{r}_{k+1}\end{bmatrix}}_{\mathbf{x}_{k+1}} = \underbrace{\begin{bmatrix}\mathbf{p}_k + \Delta t(-\hat{\boldsymbol{\omega}}_k\ \mathbf{p}_k - \mathbf{v}_k)\\ \mathbf{v}_k + \Delta t(-\hat{\boldsymbol{\omega}}_k\ \mathbf{v}_k + \mathbf{a}_k + \mathbf{g}_k)\\ \mathbf{p}_{obs,k} + \Delta t(-\hat{\boldsymbol{\omega}}_k\ \mathbf{p}_{obs,k} - \mathbf{v}_k + \mathbf{v}_{obs,k})\\ \mathbf{v}_{obs,k} + \Delta t(-\hat{\boldsymbol{\omega}}_k\ \mathbf{v}_{obs,k} + \mathbf{g}_k)\\ \mathbf{Exp}(-\Delta t\boldsymbol{\omega}_k)\ \mathbf{g}_k\\ \mathbf{Exp}(-\Delta t\boldsymbol{\omega}_k)\ \mathbf{r}_k\end{bmatrix}}_{\mathbf{x}_k \oplus (\Delta t\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k))} \tag{10}$$

where the state manifold is $\mathcal{M} = \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{S}^2 \times \mathbb{S}^2$, $dim(\mathcal{M}) = 16$, the $\oplus$ is defined accordingly on $\mathcal{M}$, and

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix}-\hat{\boldsymbol{\omega}}\mathbf{p} - \mathbf{v}\\ -\hat{\boldsymbol{\omega}}\mathbf{v} - a_T\mathbf{e}_3 + \mathbf{g}\\ -\hat{\boldsymbol{\omega}}\mathbf{p}_{obs} - (\mathbf{v} - \mathbf{v}_{obs})\\ -\hat{\boldsymbol{\omega}}\mathbf{v}_{obs} + \mathbf{g}\\ -\boldsymbol{\omega}\\ -\boldsymbol{\omega}\end{bmatrix} \in \mathbb{R}^{18}. \tag{11}$$

To sum up, for both the hovering and obstacle avoidance task, the system can be represented in the following canonical form

$$\mathbf{x}_{k+1} = \mathbf{x}_k \oplus (\Delta t\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)), \mathbf{x} \in \mathcal{M}, dim(\mathcal{M}) = n. \tag{12}$$

where the state $\mathbf{x}$ naturally evolves on the state manifold $\mathcal{M}$ with dimension $n = dim(\mathcal{M})$.

The canonical representation for both tasks allows to implement the state estimation and MPC control in the same pipeline and structure without from-scratch re-designs for each specific task or system. The general pipeline for on-manifold extended Kalman filter used in this paper is detailed in our previous work [37]. For the MPC control, the design and implementation are detailed as follows.

C. On-Manifold iterative MPC

With the discrete system defined on manifold and the canonical form of the system state (12), we formulate the control problem of the system using the MPC formulation on manifold as follows:

$$\begin{aligned}&\min_{\mathbf{u}_0,\cdots,\mathbf{u}_{N-1}} C = \sum_{k=0}^{N-1} C_k(\mathbf{x}_k, \mathbf{u}_k) + C_N(\mathbf{x}_N)\\ &s.t.\ \mathbf{x}_{k+1} = \mathbf{x}_k \oplus (\Delta t\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)),\ \mathbf{x}_0 = \mathbf{x}_{init}\\ &\mathbf{u}_{min} \le \mathbf{u}_k \le \mathbf{u}_{max},\ k = 0, \cdots, N-1\end{aligned} \tag{13}$$

The definition of the cost function depends on tasks and will be specified in Section III-D for the hovering task and Section III-E for the obstacle avoidance task, respectively. To efficiently solve the above optimization problem while naturally satisfying the manifold constraints, the cost and system dynamics are both approximated around a reference trajectory $(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k) \in \mathcal{M} \times \mathbb{R}^m$. To do so, the state error and input error are defined firstly:

$$\delta\mathbf{x}_k = \mathbf{x}_k \boxminus \bar{\mathbf{x}}_k \in \mathbb{R}^n \quad \delta\mathbf{u}_k = \mathbf{u}_k - \bar{\mathbf{u}}_k \in \mathbb{R}^m \tag{14}$$

The notation $\boxminus$ denotes the error between the predicted state $\mathbf{x}_k$ and the reference one $\bar{\mathbf{x}}_k$. This error has the same dimension n as that of the manifold $\mathcal{M}$, hence it is a minimal parameterization of the state error. As shown in [32], the operation $\boxminus$ can be chosen as any local coordinates chart. Since the local coordinates chart is expressed around each point on the reference trajectory $\bar{\mathbf{x}}_k$, the amount is usually small, thus suffering from no singularity issue. Specifically, for the hovering task (8), its state error is defined as:

$$\delta\mathbf{x} = \mathbf{x} \boxminus \bar{\mathbf{x}} = \begin{bmatrix}\delta\mathbf{p}\\ \delta\mathbf{v}\\ \delta\mathbf{g}\\ \delta\mathbf{r}\end{bmatrix} = \begin{bmatrix}\mathbf{p} - \bar{\mathbf{p}}\\ \mathbf{v} - \bar{\mathbf{v}}\\ \mathbf{B}(\bar{\mathbf{g}})^T\left(\theta_{\mathbf{g}}\frac{\hat{\mathbf{g}}\mathbf{g}}{\|\hat{\mathbf{g}}\mathbf{g}\|}\right)\\ \mathbf{B}(\bar{\mathbf{r}})^T\left(\theta_{\mathbf{r}}\frac{\hat{\mathbf{r}}\mathbf{r}}{\|\hat{\mathbf{r}}\mathbf{r}\|}\right)\end{bmatrix} \in \mathbb{R}^{10} \tag{15}$$

where $\mathbf{B}(\mathbf{x}) = [\mathbf{b}_1, \mathbf{b}_2]$ consists of two orthonormal vectors that are perpendicular to $\mathbf{x} \in \mathbb{S}^2$ and $\theta_{\mathbf{x}}$ is the rotation angle from $\mathbf{x}$ to $\bar{\mathbf{x}}$. The detailed representation of $\mathbf{B}(\mathbf{x})$ function and θ can be referred in [37]. For the obstacle avoidance task (10), its state error is defined as:

$$\delta\mathbf{x} = \mathbf{x} \boxminus \bar{\mathbf{x}} = \begin{bmatrix}\delta\mathbf{p}\\ \delta\mathbf{v}\\ \delta\mathbf{p}_{obs}\\ \delta\mathbf{v}_{obs}\\ \delta\mathbf{g}\\ \delta\mathbf{r}\end{bmatrix} = \begin{bmatrix}\mathbf{p} - \bar{\mathbf{p}}\\ \mathbf{v} - \bar{\mathbf{v}}\\ \mathbf{p}_{obs} - \bar{\mathbf{p}}_{obs}\\ \mathbf{v}_{obs} - \bar{\mathbf{v}}_{obs}\\ \mathbf{B}(\bar{\mathbf{g}})^T\left(\theta_{\mathbf{g}}\frac{\hat{\mathbf{g}}\mathbf{g}}{\|\hat{\mathbf{g}}\mathbf{g}\|}\right)\\ \mathbf{B}(\bar{\mathbf{r}})^T\left(\theta_{\mathbf{r}}\frac{\hat{\mathbf{r}}\mathbf{r}}{\|\hat{\mathbf{r}}\mathbf{r}\|}\right)\end{bmatrix} \in \mathbb{R}^{16} \tag{16}$$

The inverse operator $\boxplus$ can be defined accordingly such that

$\bar{\mathbf{x}} \boxplus (\mathbf{x} \boxminus \bar{\mathbf{x}}) = \mathbf{x}, \forall \mathbf{x}, \bar{\mathbf{x}} \in \mathcal{M}$. Then, based on (14), we have

$$\mathbf{x}_k = \bar{\mathbf{x}}_k \boxplus \delta\mathbf{x}_k \in \mathcal{M} \quad \mathbf{u}_k = \bar{\mathbf{u}}_k + \delta\mathbf{u}_k \in \mathbb{R}^m \tag{17}$$

and the cost in (13) can be approximated as:

$$\begin{aligned} C =& \sum_{k=0}^{N-1} C_k\left(\bar{\mathbf{x}}_k \boxplus \delta\mathbf{x}_k, \bar{\mathbf{u}}_k + \delta\mathbf{u}_k\right) + C_N\left(\bar{\mathbf{x}}_N \boxplus \delta\mathbf{x}_N\right) \\ \approx& \sum_{k=0}^{N-1} \left(C_k\left(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k\right) + \mathbf{J}_k^T \begin{bmatrix}\delta\mathbf{x}_k \\ \delta\mathbf{u}_k\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\delta\mathbf{x}_k \\ \delta\mathbf{u}_k\end{bmatrix}^T \mathbf{H}_k \begin{bmatrix}\delta\mathbf{x}_k \\ \delta\mathbf{u}_k\end{bmatrix} \right) \\ &+ C_N\left(\bar{\mathbf{x}}_N\right) + \mathbf{J}_N^T \delta\mathbf{x}_N + \frac{1}{2}\delta\mathbf{x}_N^T \mathbf{H}_N \delta\mathbf{x}_N \end{aligned} \tag{18}$$

where $\mathbf{J}_k$ and $\mathbf{H}_k$ are the normal Jacobian and Hessian of the cost function C_k over the vector $(\delta\mathbf{x}_k, \delta\mathbf{u}_k) \in \mathbb{R}^{n+m}$.

Similarly, the system (12) is also linearized at the reference trajectory $(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)$ to obtain the linearized error system:

$$\begin{aligned} \delta\mathbf{x}_{k+1} &= \left(\mathbf{x}_k \oplus \left(\Delta t \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)\right)\right) \boxminus \left(\bar{\mathbf{x}}_k \oplus \left(\Delta t \mathbf{f}(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)\right)\right) \\ &\approx \mathbf{F}_{\bar{\mathbf{x}}_k} \delta\mathbf{x}_k + \mathbf{F}_{\bar{\mathbf{u}}_k} \delta\mathbf{u}_k \end{aligned} \tag{19}$$

where $\mathbf{F}_{\bar{\mathbf{x}}_k}$ is the partial differentiation with respect to $\delta\mathbf{x}_k$ and evaluated at zero. $\mathbf{F}_{\bar{\mathbf{u}}_k}$ is the partial differentiation with respect to $\delta\mathbf{u}_k$ and evaluated at zero. The detailed expression of them can refer to our previous work [32]. By approximating both the cost function and the system around the reference trajectory shown in (18) and (19), respectively, the original problem (13) reduces to:

$$\begin{aligned} \min_{\delta\mathbf{u}_0, \cdots, \delta\mathbf{u}_{N-1}} & \sum_{k=0}^{N-1} \left({\mathbf{J}_k}^T \begin{bmatrix}\delta\mathbf{x}_k \\ \delta\mathbf{u}_k\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\delta\mathbf{x}_k \\ \delta\mathbf{u}_k\end{bmatrix}^T \mathbf{H}_k \begin{bmatrix}\delta\mathbf{x}_k \\ \delta\mathbf{u}_k\end{bmatrix} \right) \\ & + \mathbf{J}_N^T \delta\mathbf{x}_N + \frac{1}{2}\delta\mathbf{x}_N^T \mathbf{H}_N \delta\mathbf{x}_N \\ s.t. \quad & \delta\mathbf{x}_{k+1} = \mathbf{F}_{\bar{\mathbf{x}}_k}\delta\mathbf{x}_k + \mathbf{F}_{\bar{\mathbf{u}}_k}\delta\mathbf{u}_k, \quad \delta\mathbf{x}_0 = \delta\mathbf{x}_{init} \\ & \mathbf{u}_{min} - \bar{\mathbf{u}}_k \le \delta\mathbf{u}_k \le \mathbf{u}_{max} - \bar{\mathbf{u}}_k, \ k = 0, \cdots, N-1 \end{aligned} \tag{20}$$

which can be transformed into a standard QP problem (see Sec. IV of the supplementary [34]) and solved efficiently by QP solvers.

Due to the approximation error in the cost (18) and linearization error in model (19), the optimal solution to (20) is not necessary the optimal solution to the original problem (13). To address this issue, we iterate the above linearization and solving procedure. That is, the solution $\delta\mathbf{u}_k^*$ (equivalently $\mathbf{u}_k^*$) is rolled out according to the actual system (12) to obtain an updated trajectory, which is then used to approximate the optimization problem in (13) as the QP problem (20). This procedure is repeated until convergence (i.e., $\|\delta\mathbf{u}^*\| < \delta, \forall k$) or the maximum iteration number N_{iter} is reached. The complete algorithm is summarized in Sec. III of the supplementary material [34].

D. Hover

In the hovering task, the control goal is to drive the quadrotor from any initial state $\mathbf{x}_0 = (\mathbf{p}_0, \mathbf{v}_0, \mathbf{g}_0, \mathbf{r}_0)$ to a hover state $\mathbf{x}_h = (\mathbf{p}_h, \mathbf{0}, g\mathbf{e}_3, \mathbf{r}_h)$ and stabilize the quadrotor there. The corresponding control at the hovering is $\mathbf{u}_h = (\mathbf{0}, g)$ according to Theorem 1. Therefore, $(\mathbf{x}_h, \mathbf{u}_h)$ is naturally a feasible trajectory satisfying (12).

In order to ensure $\mathbf{r}_h$ fulfill $g\mathbf{e}_3^T\mathbf{r}_h = c$ (see (3)), we calculate the constant c according to the current state $c = \mathbf{g}_0^T\mathbf{r}_0$. Then, $\mathbf{r}_h$ can be chosen as:

$$\mathbf{r}_h = \begin{bmatrix}\sqrt{1-(c/g)^2}\sin\theta & \sqrt{1-(c/g)^2}\cos\theta & c/g\end{bmatrix}^T \tag{21}$$

where θ corresponds to the quadrotor heading angle. We set $\theta = 0$, meaning that the heading direction is perpendicular to the plane formed by the gravity and the reference direction.

The hovering task is achieved by our iterative MPC, where the cost is

$$\begin{aligned} C_k(\mathbf{x}_k, \mathbf{u}_k) &= \|\mathbf{x}_k \boxminus \mathbf{x}_h\|_{\mathbf{Q}_k} + \|\mathbf{u}_k - \mathbf{u}_h\|_{\mathbf{R}_k} \\ C_N(\mathbf{x}_N) &= \|\mathbf{x}_N \boxminus \mathbf{x}_h\|_{\mathbf{P}_N} \end{aligned} \tag{22}$$

with $\mathbf{Q}_k$ and $\mathbf{R}_k$ being respectively the state penalty matrices and the input penalty matrices, $\mathbf{P}_N$ being the penalty matrix of the terminal state. $(\mathbf{x}_h, \mathbf{u}_h)$ is set as the initial trajectory in the iterative MPC solving, and the number of iteration is set to one ($N_{iter} = 1$), meaning that the linearization is always around the hovering state.

E. Obstacle Avoidance

To enable the quadrotor to evade a suddenly appearing obstacle from an unknown direction when hovering, an obstacle cost is defined and minimized in the iMPC according to the relative distance between the quadrotor and the obstacle:

$$C_{obs} = \|\max(d^2 - \|\mathbf{p}_{obs}\|^2, 0)\|_{Q_{obs}} \tag{23}$$

where d denotes the safe distance and Q_{obs} denotes the weight penalizing any distance below the safe distance d. As can be seen, only when the obstacle falls within the safe distance from quadrotor, the cost is activated $C_{obs} > 0$ and it grows quadratically as the obstacle approaches. In the avoidance process, to prevent violent actions that may destabilize the quadrotor, it is required to keep close to the hover state simultaneously. Therefore, the cost in (13) for this task can be defined as:

$$\begin{aligned} C_k &= \|\mathbf{x}_k \boxminus \mathbf{x}_h\|_{\mathbf{Q}_k} + \|\mathbf{u}_k - \mathbf{u}_h\|_{\mathbf{R}_k} + \|\max(d^2 - \|\mathbf{p}_{obs,k}\|^2, 0)\|_{Q_{obs}} \\ C_N &= \|\mathbf{x}_N \boxminus \mathbf{x}_h\|_{\mathbf{P}_N} + \|\max(d^2 - \|\mathbf{p}_{obs,N}\|^2, 0)\|_{Q_{obs}} \end{aligned} \tag{24}$$

In this task, system state $\mathbf{x}$ (and hence $\mathbf{x}_h$) also includes obstacle state $(\mathbf{p}_{obs}, \mathbf{v}_{obs})$. Since the obstacle position does not affect the hover after the quadrotor evades it, we set the penalty matrix $\mathbf{Q}_k$ and $\mathbf{P}_N$ that are associated to the components $\mathbf{p}_{obs}, \mathbf{v}_{obs}$ to be zeroes. Moreover, we roll out the control action $\mathbf{u}_h = (\mathbf{0}, g)$ to obtain $\mathbf{x}_h$, which is used as the initial trajectory in the iterative MPC solving.

IV. Simulations and Experiments

In this section, we present simulation and experiment results to verify the proposed robocentric model method and the designed estimator and controller. In the simulation, our method is compared with a recent IBVS work on quadrotor hovering task [6]. In the experiment, hovering task and obstacle avoidance task are both conducted.

A. Simulation results

For a fair comparison, our method is applied to the same task used in [6], where the UAV starts from an initial position $[-1, -0.6, -3]$ in the world frame and aims to reach a desired hover position at $[0, 0, -1]$ by observing four co-planar visual targets (see Fig. 3). The system setup, state estimation, controller and all parameters of each method are chosen to make the comparison as fair as possible and their details are provided in Sec. V in our supplementary material [34].

The simulation results are presented from Figures 3 to 5. To compare the two methods in one criterion, the response trajectories are colored in time and displayed in Fig. 3 and the UAV position $\boldsymbol{\zeta}$ in world frame versus time is shown in Fig. 4. As can be seen, it consumes 3.8 s for our method to reach within 0.2 m to the desired position while IBVS method uses 5.6 s. Meanwhile, the control inputs (i.e., angular velocity $\boldsymbol{\omega}$ and collective thrust acceleration a_T) are guaranteed to be within the same limits as shown in Fig. 5. These results suggest that our method has faster response than the IBVS method while admitting the same bound for control efforts. The response improvement of our method is mainly attributed to the elimination of additional attitude controller, which introduces further response delay when executing the visual servoing control commands. In contrast, our visual servoing controller directly actuates the angular velocity command that has a lower response delay and hence exploits more aggressive motions. Finally, we should be noted that incorporating angular velocity control into IBVS methods is non-trivial because the image moments in IBVS methods are typically specially designed to ensure a certain invariance with respect to the control action. Taking angular velocity as the control action will often violate these invarance properties.

B. Hardware Setup

The experiments are conducted on a Q250 quadrotor airframe equipped with an onboard computer DJI Manifold-2C (Intel

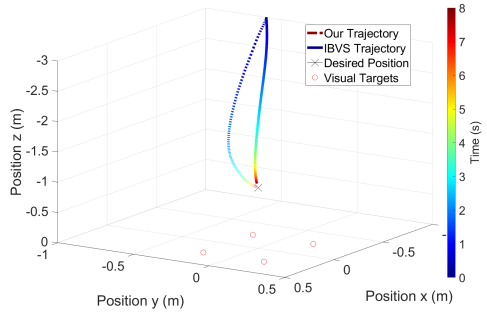


Fig. 3. Trajectory of our method and the IBVS method [6].

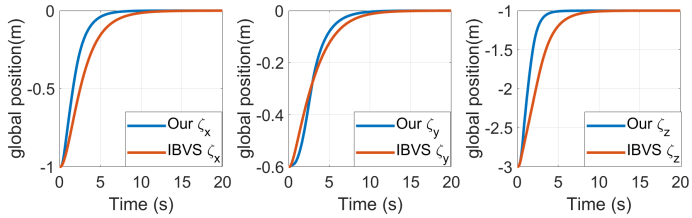


Fig. 4. UAV position $\boldsymbol{\zeta}$ of our method and the IBVS method [6].

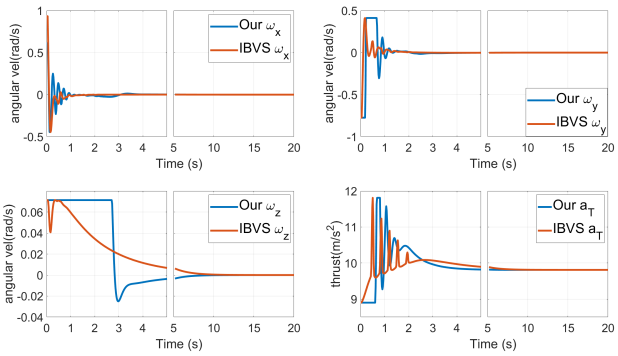


Fig. 5. Control inputs of our method and the IBVS method [6].

Core i7-8550U @1.80GHz with 8 cores) for image processing, state estimation and visual servoing control, a flight controller Pixhawk4 mini for low-level control (angular velocity control and collective thrust acceleration mixing), and an Intel Realsense ($87° \times 58°$ FoV and 0.33 m to 6 m range) for detection of visual targets. The quadrotor and all its components are showed in Fig. 6.

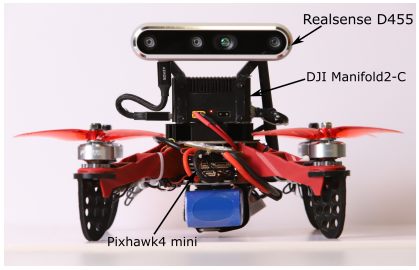


Fig. 6. The quadrotor used in our experiments.

C. Detection of visual targets

Tennis balls are used as targets in the experiments. Our method doesn't restrict the shape of targets to ball. It is used in the experiment due to the convenience and robustness of detection for circles. The detection algorithm leverages the SimpleBlobDetector function in OpenCV library to track circles appearing in each image. Prior knowledge of the ball size is not needed.

One or two balls are hung on a bracket as static targets. After they are detected, each ball is tracked based on its radius and pixel location in the last frame. Once a third ball appears in the camera field of view, it is identified as a dynamic obstacle. In our algorithm, center position of the detected circles is used as the target position in the camera frame. The detection code runs on the onboard computer at 40 Hz. The positions of static targets or obstacle ones in the camera frame are then transformed to the body frame based on the known extrinsic.

The perception and all other modules including state estimation and control are coded in C++ language and communicated with each other using ROS Kinetic.

D. State Estimation Tests

As detailed in Section III, our proposed visual servoing method performs integrated state estimation and control based on the same robocentric model. To verify the effectiveness of

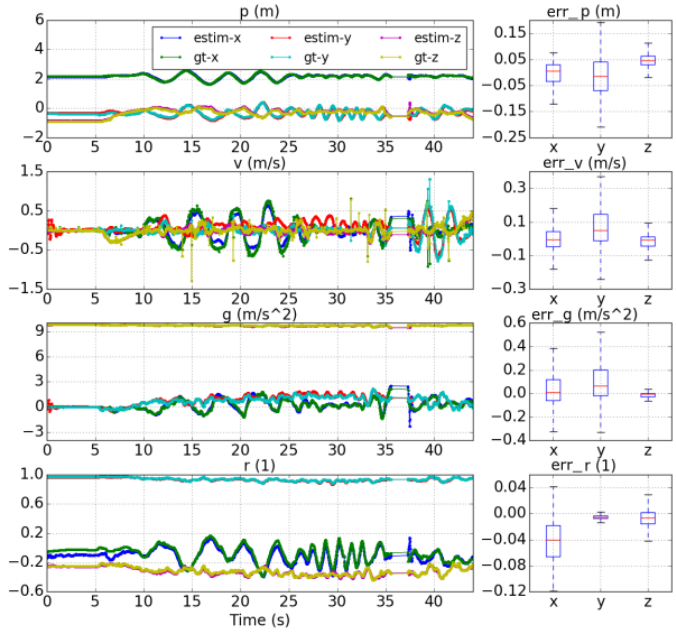


Fig. 7. The estimated (estim-) and ground true (gt-) states and the estimation error distribution.

the state estimation and target detection module, we handheld the quadrotor UAV to move randomly in front of two visual targets and examine the estimation of the state $[\mathbf{p}, \mathbf{v}, \mathbf{g}, \mathbf{r}]$ in (6) against the ground-truth measured by an external motion capture system. The estimated and ground-true states are shown in Fig. 7, where a temporary data loss occured during 35 s and 37 s. It can be seen that, with the detected visual targets, the states estimated by our designed estimator converge to the ground truth values quickly, which agrees with our analysis that the system is observable. The estimation errors after convergence are very small (a few centimeters for position). As shown in Fig. 8, the estimated IMU bias $\mathbf{b}_g$ and $\mathbf{b}_a$ also converge to stable values within the normal range of typical IMU biases as the system is excited by more aggressive motions.

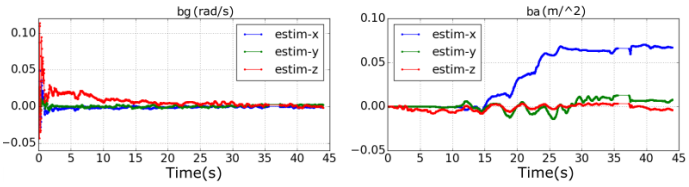


Fig. 8. Estimation of IMU bias $\mathbf{b}_g$ and $\mathbf{b}_a$.

E. Hover Test

We first apply our robocentric visual servoing method to the hovering task. The controller runs at 50 Hz on onboard computer and the sampling interval of MPC is 0.02 s. The prediction horizon N is set to 12. The penalty matrices of cost (22) are chosen as $\mathbf{Q}_k = \text{diag}([125, 40, 45, 80, 30, 10, 10, 10, 180, 10])$, $\mathbf{R}_k = \text{diag}([1, 2, 1, 0.2])$ and $\mathbf{P}_N = \mathbf{Q}_k$. The input constraints are $\mathbf{u}_{max} = [3.5, 3.5, 1.75, 12.0]^T$ and $\mathbf{u}_{min} = [-3.5, -3.5, -1.75, 5.0]^T$, respectively. The quadrotor is launched from ground by a human pilot manually, and then stabilized near hovering in front of the static visual targets. Once the camera detects the visual targets, their initial positions and the reference direction are recorded as $\mathbf{p}_0$ and $\mathbf{r}_0$, respectively. Then, the desired hovering position of the quadrotor is set to $\mathbf{p}_h = \mathbf{p}_0 - 0.5\mathbf{e}_1$ where the displacement 0.5 m along the x direction is intentionally added to excite the quadrotor. The desired reference direction is set to $\mathbf{r}_h = \left(0, \sqrt{1 - \left(\mathbf{g}_0^T \mathbf{r}_0 / g\right)^2}, \mathbf{g}_0^T \mathbf{r}_0 / g\right)$ in consideration of the constraint (21). Finally, the maximum iteration number N_{iter} is set to 1 as the hovering state trajectory $\mathbf{x}_h$ is pre-known (see Theorem 1), requiring no trajectory iteration.

First, we test the hovering task with only one visual target (see model (1)). The results of the control error are shown in Fig. 9. As can be seen, from the initial position, the quadrotor successfully converges to the desired hovering position (with error below 0.07 m) after 2.5 s. After convergence, the position error keeps below 0.07 m and the velocity keeps below 0.1 m/s, which achieves a stable hovering.

As discussed before, with only one visual target, the quadrotor cannot hover at a unique point. This phenomenon is apparent in a third person camera view shown in Fig. 10. The figure shows the quadrotor in the fixed camera view drifts from right to left from 54 s to 60 s. It can be seen that, while the relative position to the target are both small (below 0.07 m, see Fig. 9) during this period, the quadrotor drifts freely on a horizontal circle (see Fig. 10).

Then, we test the hovering task with two visual targets (see model (2)). Compared to the previous condition where only one target exists, the quadrotor heading direction (i.e., orientation) is observable and controllable when two targets exist. To show this, in the experiment, after the quadrotor hovers at a desired position, we use a pole to poke the quadrotor to apply a consistent lateral disturbance as shown in Fig. 11. The direction error shown in Fig. 12 denotes the two components of $\delta\mathbf{r} = \mathbf{r} \boxminus \mathbf{r_h} \in \mathbb{R}^2$, whose norm $\|\delta\mathbf{r}\|$ represents the quadrotor orientation error in radians. With the consistent lateral disturbance, the position error reaches 0.45 m while the orientation error $\|\delta\mathbf{r}\|$ reaches around 0.28 rad (see Fig. 11(a) and Fig. 12). After the disturbance is released, the quadrotor recovers to its initial hovering state (position error below 0.15 m, orientation error below 0.05 rad) after 8.6 s (see Fig. 11(b) and Fig. 12). To

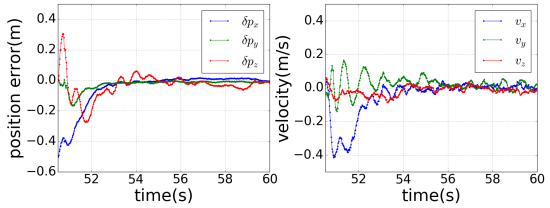


Fig. 9. Convergence of the position error in the hovering task.

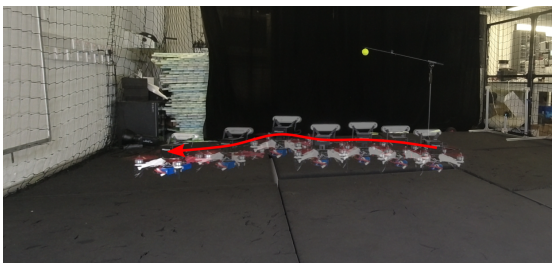


Fig. 10. The quadrotor is free to drift along a horizontal circle.

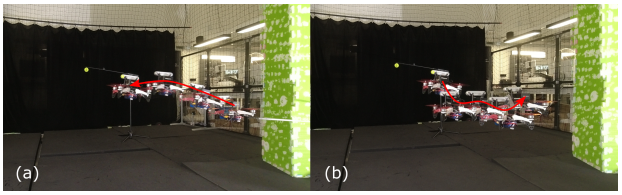


Fig. 11. (a) Perturbed by a consistent lateral disturbance, the quadrotor moves away from initial position. (b) The quadrotor recovers to the initial position and orientation.

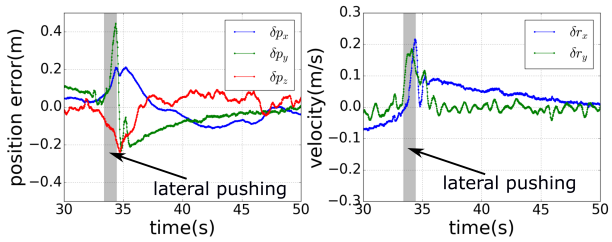


Fig. 12. Time evolution of position error and direction error in the presence of a consistent lateral disturbance (the shaded area denotes the period during which the lateral disturbance is applied).

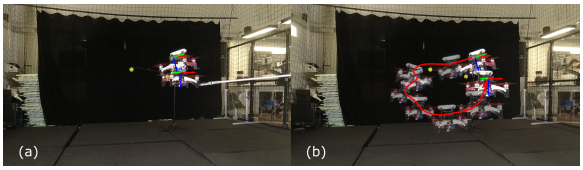


Fig. 13. (a) The quadrotor encounters a sudden rotation when one of its landing gear is stabbed (b) The qudrotor recovers to its initial hovering state.

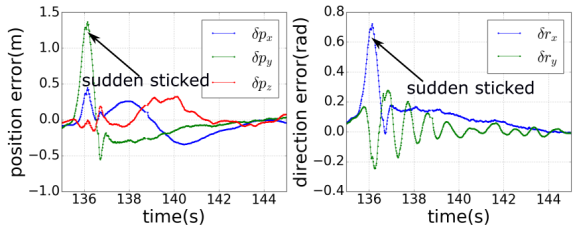


Fig. 14. Time evolution of position and direction error when the quadrotor encounters a sudden rotation.

sum up, with an additional target to form a direction with the first target, the quadrotor is able to hover at a desired position without drift.

Finally, we verify the disturbance rejection of the proposed robocentric visual servoing controller by applying a sudden rotation and a consistent forward push as shown in Fig. 13 and Fig. 15. In case of the sudden rotation, the position error caused by the disturbance is up to 1.35 m on the y axis and the orientation error is up to 0.7 rad as shown in Fig. 14. The quadrotor recovers to its initial hovering state (position error below 0.15 m, orientation error below 0.05 rad) after 6.1 s. In case of the consistent forward pushing, the maximum position error along body x axis is 0.42 m while the orientation is only slightly perturbed as shown in Fig. 16. Once releasing the forward push, the quadrotor recovers to its hovering state (position error below 0.15 m, orientation error below 0.05 rad) in 1.9 s. Several more similar experiments are conducted and their results are presented in Sec. VI.1 in the supplementary material [34]. It can be concluded from above experiments that the system shows high ability to resist external disturbances.

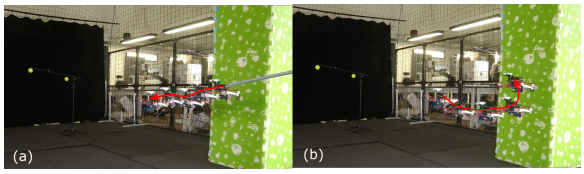


Fig. 15. (a) The quadrotor is consistently pushed away from its initial position (b) It immediately recovers to the initial position when the push force is released.

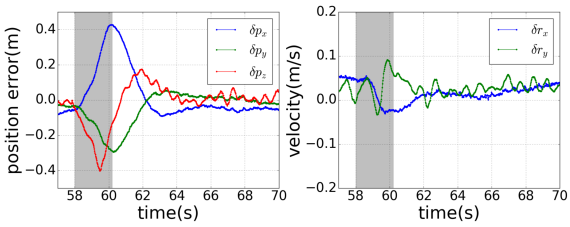


Fig. 16. Time evolution of position and direction error when the quadrotor is perturbed by a consistent forward push (the shaded area denotes the period during which the push is active).

In terms of computation time, the solving time for the iMPC controller in the above experiments is within $[0.0049, 0.0096]$ s and the average value is 0.0064 s, which are all below the MPC running period 0.02 s.

F. Obstacle Avoidance Test

Our proposed visual servoing formulation is used to avoid a thrown ball identified as a dynamic obstacle. The iMPC is implemented for both planning and tracking control. The safe distance d in (23) is set to 2 m. The penalty matrices of cost (24) are chosen as $\mathbf{Q}_k = \text{diag}([80, 30, 30, 30, 5, 5, 10, 10, 40, 15])$, $\mathbf{R}_k = \text{diag}([0.6, 0.6, 0.4, 0.1])$ and $\mathbf{P}_N = \mathbf{Q}_k$. The penalty factor is $Q_{obs} = 600$. The input constraints are $\mathbf{u}_{max} = [7, 7, 5, 16.0]^T$ and $\mathbf{u}_{min} = [-7, -7, -5, 2.0]^T$ respectively. The maximum iteration number N_{iter} is set to 3. To mitigate the increased computation load caused by more iterations, we set the prediction horizons N to 4, a shorter value than that in the hovering task. As a result, the planning and control rate of our iMPC remains at 50 Hz.

An example of the obstacle avoidance experiments is shown in Fig. 17. Before the ball is thrown out, the obstacle cost (23) is always equal to zero and the quadrotor hovers at the desired position. Once the ball appears in the camera FoV and is within the safe distance, it is identified as an obstacle and its trajectory within the prediction horizon is predicted according to the dynamic model (4). At the same time, C_{obs} grows as the obstacle approaches the quadrotor. During this process, the static targets are still detected and the avoidance action causes a considerable position error from the hover position. After a successful avoidance, the quadrotor recovers to hover in front of the static targets.

The quadrotor position relative to the hovering position and the detected obstacle distance and speed are shown in Fig. 18(a) and Fig. 18(b) respectively. As can be seen, the ball is thrown out and detected by the quadrotor at around 54.4 s, at a 2.5 m distance. At 54.6 s, the ball falls within the safe distance (i.e., 2 m) and the quadrotor starts to react as depicted in Fig. 18(a) (also see Fig. 17(a)). The evading maneuver takes 0.5 s (from 54.6 s to 55.1 s, see Fig. 17(d)), after which the

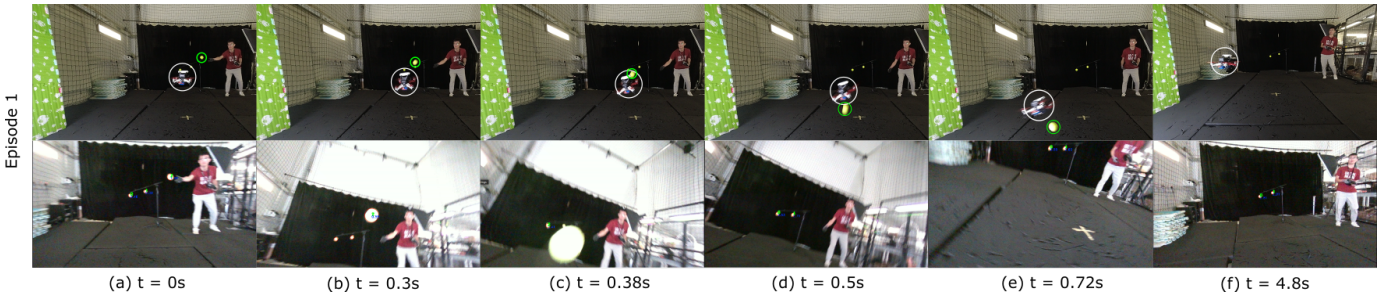
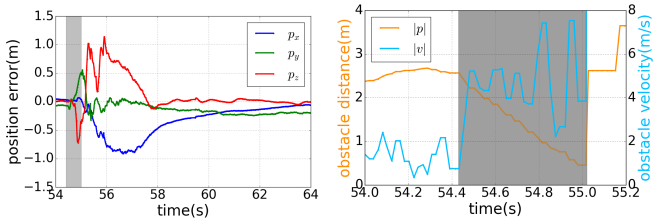


Fig. 17. Episode 1: avoid an incoming ball by moving backward. The relative velocity is up to 7.4 m/s. The upper images capture the avoidance sequence from an external camera and the lower images are respective views from the onboard camera. (a) the ball is detected within the safe distance. (b) the quadrotor pitches up to move backward. (c) the closest relative distance. (d) successfully avoid the obstacle. (e) the quadrotor begins to recover to the hovering state. (f) back to hovering.

quadrotor recovers to the initial hovering position (with error below 0.3 m) at 59.4 s (see Fig. 17 (f)). The whole process from detecting the ball, avoiding it to recovering back to hover lasts 4.8 s (from 54.6 s to 59.4 s). The maximum estimated relative velocity between the ball and the quadrotor is up to 7.4 m/s as shown in Fig. 18(b) and the quadrotor is allowed to react when the ball is within a close distance (i.e., 2 m), both making the avoidance very challenging. The quadrotor moves up to 1.3 m from the hovering position at 55.9 s to avoid the obstacle. The average solving time of iMPC in the avoidance process is 0.018 s which is below the running period 0.02 s of the iMPC.



(a) quadrotor's position error (b) obstacle's position and velocity

Fig. 18. (a) The position error when the quadrotor is reacting to avoid the dynamic obstacle. (b) The meausured obstacle distance and estimated obstacle velocity. The shaded area denotes the period when the ball is thrown and approaching the quadrotor. After 55 s, the obstacle has been avoided by the quadrotor and falls out of the sensor FoV.

We present another two experiments in Sec. VI.2 of the supplementary material [34] in which the quadrotor moves forward and sideward to avoid the obstacle respectively. The quadrotor successfully avoids the balls and recovers to the original hovering position within 3.93 s and 2.68 s respectively. All the three episodes and several more experiments are presented in the video at https://youtu.be/iAODWE3eTCo.

Thrown balls are chosen as dynamic obstacles in many previous works such as [10], [11], which are both based on non-visual servoing method, due to its high speed and the quite random thrown direction, height and distance. It creates a challenging and realistic scenario for the quadrotor to testify the avoidance algorithm. In order to demonstrate the agility of the system in our experiments, the quadrotor is set to react the approaching ball at a close distance with high speed, which are 2 m and 7.4 m/s, respectively, in episode 1. The reaction time left for the quadrotor to finish the avoidance action is 0.27 s. In [10], a basketball is thrown out from around 3 m away with the relative velocity up to 10 m/s. The reaction time is 0.3 s which is comparable with our result although their work uses event camera which mainly addresses obstacle detection problem. The reaction time is longer than 0.5 s in another work [11].

V. Conclusion and Future Works

In this paper, we propose a robocentric formulation for visual servoing control. The measurements of the visual targets are directly used for state estimation and control. This formulation is implemented on the quadrotor for hover and obstacle avoidance. The controllability and observability analysis of the robocentric model are discussed. An on-manifold iterative MPC is proposed for trajectory planning and tracking. Simulations and experiments are then conducted to validate the robocentric model and iterative MPC. In the hover tests, the quadrotor accurately hovers at a desired position with small control errors. Even after significant disturbances, the quadrotor is able to stabilize itself and recover to the original hover position. In obstacle avoidance tests, the quadrotor successfully avoids tennis balls in high speed.

In future works, we could apply the robocentric formulation and iterative MPC for more challenging tasks and general scenarios, such as the consideration of camera FoV constraints in planning and control and detection of more natural visual targets other than those with specified shape or color. For the latter, we could adopt visual features (e.g., corner points) that have been widely used in feature-based visual SLAM as the visual targets. These visual features are abundant in feature-dense environments and could be matched using their descriptors (e.g., ORB descriptor). How to exploit these abundant visual features in the visual servoing framework would be an interesting future work.

References

[1] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[2] ——, "Visual servo control. ii. advanced approaches [tutorial]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007.

[3] C. Doignon and R. Stolkin, "An introduction to model-based pose estimation and 3-d tracking techniques," *Scene Reconstruction, Pose Estimation and Tracking*, vol. 530, 2007.

[4] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE robotics & automation magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[5] D. Zheng, H. Wang, J. Wang, S. Chen, W. Chen, and X. Liang, "Image-based visual servoing of a quadrotor using virtual camera approach," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 2, pp. 972–982, 2016.

[6] K. Zhang, Y. Shi, and H. Sheng, "Robust nonlinear model predictive control based visual servoing of quadrotor uavs," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 2, pp. 700–708, 2021.

[7] D. Guo and K. K. Leang, "Image-based estimation, planning, and control for high-speed flying through multiple openings," *The International Journal of Robotics Research*, vol. 39, no. 9, pp. 1122–1137, 2020.

[8] P. Serra, R. Cunha, T. Hamel, D. Cabecinhas, and C. Silvestre, "Landing of a quadrotor on a moving target using dynamic image-based visual servo control," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1524–1535, 2016.

[9] J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar, "Visual servoing of quadrotors for perching by hanging from cylindrical objects," *IEEE robotics and automation letters*, vol. 1, no. 1, pp. 57–64, 2015.

[10] D. Falanga, K. Kleber, and D. Scaramuzza, "Dynamic obstacle avoidance for quadrotors with event cameras," *Science Robotics*, vol. 5, no. 40, 2020.

[11] B. Lindqvist, S. S. Mansouri, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "Nonlinear mpc for collision avoidance and control of uavs with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6001–6008, 2020.

[12] J. Tordesillas and J. P. How, "Panther: Perception-aware trajectory planner in dynamic environments," *arXiv preprint arXiv:2103.06372*, 2021.

[13] J. Lin, H. Zhu, and J. Alonso-Mora, "Robust vision-based obstacle avoidance for micro aerial vehicles in dynamic environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2682–2688.

[14] F. Kong, W. Xu, and F. Zhang, "Avoiding dynamic small obstacles with onboard sensing and computating on aerial robots," *arXiv preprint arXiv:2103.00406*, 2021.

[15] D. Lee, H. Lim, and H. J. Kim, "Obstacle avoidance using image-based visual servoing integrated with nonlinear model predictive control," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 5689–5694.

[16] A. Mcfadyen, L. Mejias, P. Corke, and C. Pradalier, "Aircraft collision avoidance using spherical visual predictive control and single point features," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 50–56.

[17] Z. Tang, R. Cunha, D. Cabecinhas, T. Hamel, and C. Silvestre, "Quadrotor going through a window and landing: An image-based visual servo control approach," *Control Engineering Practice*, vol. 112, p. 104827, 2021.

[18] G. Rigatos, P. Siano, D. Selisteanu, and R. Precup, "Nonlinear optimal control of oxygen and carbon dioxide levels in blood," *Intelligent Industrial Systems*, vol. 3, no. 2, pp. 61–75, 2017.

[19] Y. Li, Y. Qin, W. Xu, and F. Zhang, "Modeling, identification, and control of non-minimum phase dynamics of bi-copter uavs," in *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2020, pp. 1249–1255.

[20] T. Chen, A. Babanin, A. Muhammad, C. Bert, and C. Cyj, "Modified evolved bat algorithm of fuzzy optimal control for complex nonlinear systems," *Rom. J. Inf. Sci. Technol*, vol. 23, pp. T28–T40, 2020.

[21] A. Sarabakha and E. Kayacan, "Online deep fuzzy learning for control of nonlinear systems using expert knowledge," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 7, pp. 1492–1503, 2019.

[22] E.-H. Zheng, J.-J. Xiong, and J.-L. Luo, "Second order sliding mode control for a quadrotor uav," *ISA transactions*, vol. 53, no. 4, pp. 1350–1356, 2014.

[23] M. Abdelaal, M. Fränzle, and A. Hahn, "Nonlinear model predictive control for trajectory tracking and collision avoidance of underactuated vessels with disturbances," *Ocean Engineering*, vol. 160, pp. 168–180, 2018.

[24] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "Pampc: Perception-aware model predictive control for quadrotors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–8.

[25] I. A. Zamfirache, R.-E. Precup, R.-C. Roman, and E. M. Petriu, "Reinforcement learning-based control using q-learning and gravitational search algorithm with experimental validation on a nonlinear servo system," *Information Sciences*, vol. 583, pp. 99–120, 2022.

[26] N. O. Lambert, D. S. Drew, J. Yaconelli, S. Levine, R. Calandra, and K. S. Pister, "Low-level control of a quadrotor with deep model-based reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4224–4230, 2019.

[27] M. W. Mehrez, G. K. I Mann, and R. G. Gosine, "An optimization based approach for relative localization and relative tracking control in multi-robot systems," *Journal of Intelligent & Robotic Systems*, vol. 85, no. 2, pp. 385–408, 2017.

[28] M. Kamel, M. Burri, and R. Siegwart, "Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463–3469, 2017.

[29] M. Sauvée, P. Poignet, E. Dombre, and E. Courtial, "Image based visual servoing through nonlinear model predictive control," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 1776–1781.

[30] J. Gao, A. A. Proctor, Y. Shi, and C. Bradley, "Hierarchical model predictive image-based visual servoing of underwater vehicles with adaptive neural network dynamic control," *IEEE transactions on cybernetics*, vol. 46, no. 10, pp. 2323–2334, 2015.

[31] H. Sheng, E. Shi, and K. Zhang, "Image-based visual servoing of a quadrotor with improved visibility using model predictive control," in *2019 IEEE 28th international symposium on industrial electronics (ISIE)*. IEEE, 2019, pp. 551–556.

[32] G. Lu, W. Xu, and F. Zhang, "On-manifold model predictive control for trajectory tracking on robotic systems," *IEEE Transactions on Industrial Electronics*, 2022.

[33] K. M. Lynch and F. C. Park, *Modern robotics*. Cambridge University Press, 2017.

[34] Y. Li, G. Lu, D. He, and F. Zhang. Supplementary material of paper: Robocentric model-based visual servoing for quadrotor flights. [Online]. Available: https://connecthkuhk-my.sharepoint.com/:b:/g/personal/yhangli_connect_hku_hk/EVTxFR3Gx_ZMiMY2gS8oL_wBVVYceGx1JAVlxouo9V5u1w?e=6E2Pvg

[35] E. J. Lefferts, F. L. Markley, and M. D. Shuster, "Kalman filtering for spacecraft attitude estimation," *Journal of Guidance, control, and Dynamics*, vol. 5, no. 5, pp. 417–429, 1982.

[36] W. Xu, D. He, Y. Cai, and F. Zhang, "Robots' state estimation and observability analysis based on statistical motion models," *IEEE Transactions on Control Systems Technology*, 2022.

[37] D. He, W. Xu, and F. Zhang, "Embedding manifold structures into kalman filters," *arXiv preprint arXiv:2102.03804*, 2021.

[38] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 2017.

Yihang Li received the B.S. degree in Mechanics from the University of Science and Technology of China (USTC), China, in 2019. He has been a Ph.D student in the Department of Mechanical Engineering, the University of Hong Kong since 2019. His research interests include UAV, control and visual servoing based navigation.

Guozheng Lu received the B.Eng. degree in Automation from Harbin Institute of Technology, Harbin, China, in 2016. He was a control engineer in Dajiang Innovations (DJI), Shenzhen, China from 2016 to 2019, and has been a Ph.D candidate in the Department of Mechanical Engineering, the University of Hong Kong since 2020. His research interests include control and trajectory generation on robotic systems.

Dongjiao He received the B.S. degree in Mechanical Engineering, in 2016, from Southeast University, Nanjing, China, and M.S. degree in Mechanical Engineering, in 2019, from Shanghai Jiao Tong University, Shanghai, China. She is currently a Ph.D. candidate in Department of Mechanical Engineering, University of Hong Kong, Hong Kong, China. Her research interests include mechatronics and robotics, with focus on sensor fusion, UAV control and lidar-based navigation.

Fu Zhang received the B.E. in Automation, in 2011, from the University of Science and Technology of China (USTC), China, and the Ph.D. degree in Mechanical Engineering, in 2015, from University of California, Berkeley, USA. He joined the University of Hong Kong, in 2018, where he is currently an assistant professor in Mechanical Engineering. His research interests include mechatronics and robotics, with focus on UAV design, control, and lidar-based navigation.