Experimental Nonrobocentric Dynamic Landing of Quadrotor UAVs With On-Ground Sensor Suite

Li-Yu Lo, *Associate Member, IEEE*, Boyang Li, Chih-Yung Wen, and Ching-Wei Chang

Abstract—**In this work, we propose a novel dynamic landing solution utilizing an on-ground sensor suite, eliminating the need for airborne exteroceptive sensors and expensive computational units. All localization and control modules operate in a noninertial frame. The system begins with a relative state estimator that tracks the unmanned aerial vehicle's (UAV) state via onboard light-emitting diode (LED) markers and an on-ground camera. The state is geometrically expressed on a manifold and estimated using an iterated extended Kalman filter (IEKF) algorithm. A motion planning module is then developed to guide the landing process, leveraging the differential flatness property to formulate it as a minimum jerk trajectory. Considering visibility and dynamic constraints, the problem is solved using quadratic programming (QP), with the final motion primitive represented through piecewise polynomials. A series of experiments validate the applicability of the proposed approach, achieving successful landings of an 18 × 18 cm quadrotor on a 43 × 43 cm platform, demonstrating performance comparable to conventional methods. In addition, comprehensive hardware and software details are provided for future reference within the research community.**

Index Terms—**Dynamic landing, motion planning, quadrotor unmanned aerial vehicle (UAV), relative pose estimation, unmanned ground vehicle (UGV).**

Nomenclature

$\mathbb{SO}(3)$	Special orthogonal group.
$\mathbb{SE}(3)$	Special Euclidean group.
$\mathcal{M}$	Manifold group.
$\mathfrak{m}$	Tangent space of a manifold.
$\mathcal{I}$	Inertial frame.
$\mathcal{N}$	Noninertial frame.
$\mathcal{B}$	Body frame.
$\mathbf{T}_{\mathcal{Y}}^{\mathcal{X}} \in \mathbb{SE}(3)$	Transformation matrix from frame $\mathcal{Y}$ to $\mathcal{X}$.
${}^{\mathcal{A}}\mathbf{r}$	Vector $\mathbf{r}$ expressed in frame $\mathcal{A}$.
$\hat{\cdot}$	Estimation.
$\bar{\cdot}$	A priori.
$\boxplus, \boxminus$	Group operators.
$\preceq, \succeq$	Generalized inequalities.

Received 7 May 2024; revised 5 August 2024; accepted 3 September 2024. Date of publication 30 September 2024; date of current version 24 October 2024. This work was supported in part by the Innovation and Technology Commission of Hong Kong under Grant ITT/027/19GP, in part by the PolyU Presidential PhD Fellowship Scheme (PPPFS), and in part by InnoHK Center of ITC (InnoHK-HKCRC). The Associate Editor coordinating the review process was Dr. Bin Zi. *(Corresponding author: Ching-Wei Chang.)*

Li-Yu Lo and Chih-Yung Wen are with AIRo-LAB, Department of Aeronautical and Aviation Engineering and the Research Centre for Unmanned Autonomous System, The Hong Kong Polytechnic University, Hong Kong (e-mail: patty.lo@connect.polyu.hk; chihyung.wen@polyu.edu.hk).

Boyang Li is with the School of Engineering, The University of Newcastle, Callaghan, NSW 2308, Australia (e-mail: Boyang.li@newcastle.edu.au).

Ching-Wei Chang is with Hong Kong Center for Construction Robotics, The Hong Kong University of Science and Technology, Hong Kong (e-mail: ccw@ust.hk).

Data is available on-line at https://youtu.be/7wiCh46MQmc

Digital Object Identifier 10.1109/TIM.2024.3470976

I. Introduction

Due to its high versatility and easy deployment, unmanned aerial systems, especially multirotor unmanned aerial vehicles (UAV) like quadrotors, have garnered significant attention in academia and industry over the past decade [1], [2]. Safe takeoff and landing capabilities are crucial for the success of aerial missions, particularly in scenarios involving frequent releasing and retracting of drones, as seen in large-scale deployments for tasks, such as marine surveying [3] or drone-assisted parcel delivery [4]. The dynamic performance of landing tasks is often essential for efficiency despite introducing challenges and potential threats to mission reliability, making it a topic worthy of investigation.

However, rather than adopting an on-board approach, we relocate most sensing and computing modules to the ground. Consequently, the airborne quadrotor is equipped solely with a low-cost flight controller unit, which we classify as a "nonrobocentric approach" in this study, where the aerial robot has no independent sensing and decision-making ability. The argument put forth is that this particular configuration has the potential to significantly reduce the payload of the UAV and simplify its avionics system, all while maintaining a performance level comparable with that of a traditional autonomous landing system. Moreover, in cases where a UAV mission does not necessitate the utilization of computationally intensive perception modules (as exemplified by Zhou et al. [3]), it is posited that the devised system could serve as a more efficient solution, low power-consuming for retracting UAVs. In addition, in numerous research studies, dynamic landing is often performed by estimating the velocity of the landing target. However, due to the inherent limitations of vision systems, velocity estimation tends to be relatively unreliable [5, p. 206], which can adversely impact the landing mission. In contrast, the proposed nonrobocentric approach enables the system to obtain comprehensive velocity information of the landing target, allowing for its full utilization during landing trajectory tracking.

Via formulating a noninertial control problem, the proposed framework relies on state-of-the-art machine vision algorithms, quadrotor motion planning, and control, all of which are executed by the sensing and computing modules situated on the ground. This approach ensures optimal performance while alleviating the computational burden on the airborne quadrotor. Aligned with our previous investigation in [6], this article endeavor approaches the problem with a refined problem formulation and introduces each submodule with finer details. Contributions are summarized as follows.

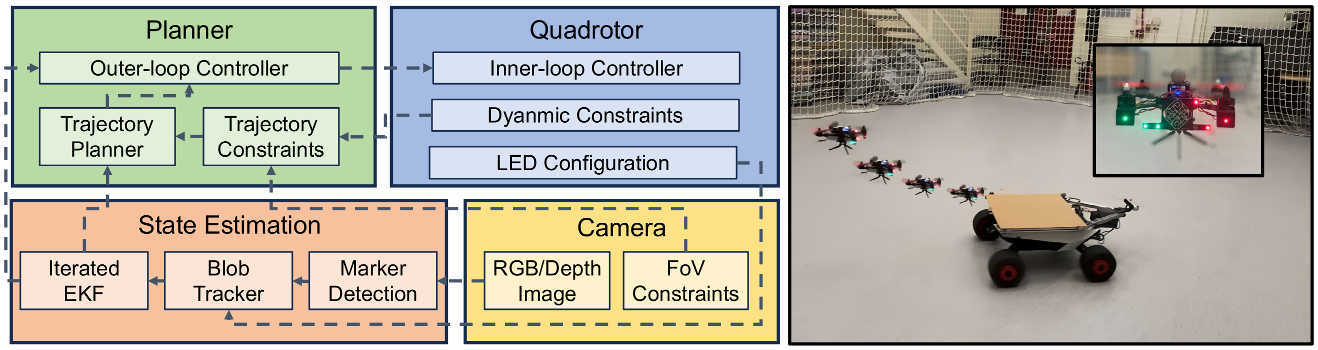


Fig. 1. Hardware and software details of the proposed nonrobocentric landing system. The system could be separately discussed in two main parts: state estimation and planner. Note that all modules above are located on the ground, with no exteroceptive sensors and computing units on the quadrotor.

1) A nonrobocentric (offboard sensing and computing) autonomous landing system is presented, along with its problem formulation.
2) A relative state estimation (RSE) module based on an iterated extended Kalman filter (IEKF) is proposed for aggressive maneuver in the noninertial control framework.
3) By considering the landing safety corridors and dynamic constraints, a method to generate a touchdown trajectory is designed to ensure the safety and smoothness of the landing process.
4) We integrate all the above-described modules and validate the performance via experiments. All hardware and software implementation details are released for future research.

The hardware and software architectures are illustrated in Fig. 1, where relations between each submodule are shown. The rest of this article is arranged as follows. Section II first mentions some significant pioneering work. Section III then describes the problem definition and the overall software architecture. Section IV further illustrates the adopted method for RSE. In Section V, the motion planning for landing is elucidated. Then, Section VI will attempt to evaluate the designed system through practical experiments. Finally, the conclusion is drawn in Section VII.

II. Related Works

In recent years, autonomous landing for unmanned aerial systems has received significant attention. Simultaneously, the emergence of low-cost computation components has presented an opportunity for the field of machine vision to expand. The aforementioned developments have, therefore, led to a vast body of literature that addresses both static and dynamic landing scenarios. Among these works, Lee et al. [7] presented an early and prominent study that employed a conventional visual servoing approach to facilitate the landing of a quadrotor on a moving platform. Later on, other visual servoing-based methods for autonomous landing could also be seen in [8], [9], [10], and [11]. Adopting an autonomous navigation framework, Falanga et al. [5] developed a comprehensive vision system for quadcopters by combining vision odometry [12] and a trajectory planning module while tracking the relative state of the landing pad with computer vision markers. With a slight difference in control approaches, Baca et al. [13] also presented a UAV system that could land on a real-life ground vehicle through trajectory generation and model predictive control, where GPS, camera, and rangefinder were used as localization and sensing modules. A similar configuration could also be seen in [14], [15], [16], [17], and [18], where the exploitation of similar vision configuration and control methods was used to address the landing problem.

Despite the effectiveness of the above work, one could observe that, in many of the discussed scenarios, other than the final landing, the overall flight missions do not necessarily require the installation of vision sensors and high-level embedded computers. For instance, in parcel delivery mentioned in [18], an accurate GPS receiver and preknown terrain information would have sufficed for the mission objectives. From an engineering perspective, although intriguing and effective outcomes have been achieved, solving the autonomous landing problem with onboard sensing might not be the optimal solution on some occasions.

Compared with onboard sensing and computing methods, a relatively limited body of literature explores ground-based guidance, primarily attributed to its increased complexity in calculating the corresponding pose. A survey conducted by Martínez et al. [19] further pointed out that despite such a system allowing more computation resources, the field of view (FoV) is narrower; in addition, mutual communication between ground and UAV is a mandatory requirement. Nevertheless, it is asserted that the former could be resolved by designing the appropriate algorithms while adopting suitable sensors; as for communication modules, all unmanned vehicles are believed to be equipped with primary communication modules such as radio frequency receivers.

The study presented in [19] developed a trinocular guidance system designed explicitly for helicopter landings. The system could acquire accurate 3-D- translations of the aircraft by employing reconstruction techniques. Building upon this concept, Kong et al. [20], [21] proposed a pan-tilt stereo camera unit capable of detecting and tracking UAVs. To enhance robustness in adverse weather conditions, vision systems that work in the infrared spectrum were utilized. Despite some promising results retrieved from the above-listed work, much of the work only dealt with translational data, which is insufficient to control a multirotor fully in the proposed nonrobocentric framework. In addition, only the static landing problem is focused. Contrarily, Ferreira et al. [22] introduced a landing guidance system tailored for dynamic situations. This system utilized deep learning-based methods to estimate the relative six degrees of freedom (DoF) pose between a fixed-wing UAV and an aircraft carrier. However, the authors only validated the system through simulation, thus necessitating further practical experiments to ascertain its real-life applicability.

Although effective solutions were seen from the above pioneer works, most literature dealt with the static landing problem. At the same time, a small portion focused on dynamic cases; physical experiments were still lacking. Hence, this research aims to demonstrate a complete ground-based landing guidance system for dynamic cases, in which real-life experiments will be conducted.

III. Problem Definition

Given a quadrotor with solely an embedded low-cost flight controller and illuminative markers, the offboard sensing and computing devices in the noninertial frame should determine the relative state information while generating a guidance trajectory to control and land the quadrotor in a nonrobocentric fashion. The described problem could be further expressed as follows.

We first define the relative states of the quadrotor in a local reference frame, where the relative state is defined on the manifold $\mathcal{M}$

$$\mathbf{x} = [\mathbf{p} \quad \mathbf{R} \quad \mathbf{v}] \in \mathcal{M}$$

where

$$\mathcal{M} = \mathbb{R}^3 \times \mathbb{SO}(3) \times \mathbb{R}^3. \tag{1}$$

In which, $\mathbf{p}$, $\mathbf{R}$, and $\mathbf{v}$, respectively, denote the translation, rotation, and linear velocity in the noninertial frame, denoted by $\mathcal{N}$; $\mathcal{M}$ is the Cartesian product of multiple manifolds. The reason we involve manifold representation is to have easier calculus handling during filtering or optimization [23]. The estimation $\hat{\mathbf{x}}$ should be subscripted with k to denote the estimation at time step k, i.e., $\hat{\mathbf{x}}_k$. From (1) and Fig. 2, $\mathbf{p}$ and $\mathbf{R}$ also form $\mathbf{T}_{\mathcal{B}}^{\mathcal{N}}$ belonging to $\mathbb{SE}(3)$, a transformation matrix from the UAV body frame $\mathcal{B}$ to the local noninertial frame $\mathcal{N}$. We can further derive the UAV pose in the inertial frame $\mathcal{I}$ with unmanned ground vehicle (UGV) poses

$$\mathbf{T}_{\mathcal{B}}^{\mathcal{N}} = \mathbf{T}_{\mathcal{I}}^{\mathcal{N}} \mathbf{T}_{\mathcal{B}}^{\mathcal{I}}. \tag{2}$$

Meanwhile, we let $\boldsymbol{\sigma}(t)$, parameterized by t, be the desired landing trajectory for the quadrotor to track. $\boldsymbol{\sigma}(t)$ comprises the translational variable $\mathbf{r}(t)$ and its derivatives, as well as the rotational variable $\boldsymbol{\theta}(t)$

$$\boldsymbol{\sigma}(t) = (\mathbf{r}(t)^T, \dot{\mathbf{r}}^T(t), \ddot{\mathbf{r}}^T(t), \boldsymbol{\theta}^T(t))^T \quad \forall t \in [T_0, T] \tag{3}$$

where T_0 and T denotes the starting and terminating time landing trajectory. $\boldsymbol{\sigma}(\boldsymbol{t})$ should also be subject to constraints for kinematic and dynamic feasibility

$$\Lambda_i \boldsymbol{\sigma}(t) \le \boldsymbol{\beta}_i \quad \forall i \in [1, 2, \dots, N] \tag{4}$$

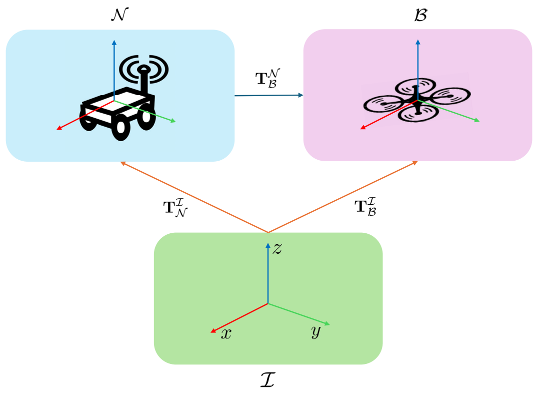

$\mathcal{N}$
$\mathcal{B}$
$\mathbf{T}_{\mathcal{B}}^{\mathcal{N}}$
$\mathbf{T}_{\mathcal{N}}^{\mathcal{I}}$
$\mathbf{T}_{\mathcal{B}}^{\mathcal{I}}$
z
x
y
$\mathcal{I}$

Fig. 2. Transformation among inertial frame $\mathcal{I}$, noninertial local target frame $\mathcal{N}$, and quadrotor body frame $\mathcal{B}$.

where Λ_i is the affine mapping matrix from $\boldsymbol{\sigma}(t)$ to constraints $\boldsymbol{\beta}_i$, where i stands for the number of constraints. The rest of this article addresses the problem of finding $\mathbf{x}_k$ for $k \in [T^-, T]$, $T^- < T_0$, while generating a constrained $\boldsymbol{\sigma}(t)$ as defined in (3) and (4) over $[T_0, T]$. All variables and constraints here are defined in $\mathcal{N}$. The proposed system then adopts the classic closed-loop control structure, where the plant receives inputs based on reference [see (3)] and estimation feedback [see (2)].

IV. Relative State Estimation

The nonrobocentric landing system starts with a robust RSE to get $\mathbf{x}_k$. Herein, a set of airborne light-emitting diode (LED) markers (as shown in Fig. 1) is attached to the quadrotor while the on-ground camera measures the relative state accordingly. The proposed module is inspired by other vision-based relative pose estimation methods [24], but with improvements in terms of robustness for highly dynamic systems such as dynamic landing in this case.

A. Marker Detection

The detection features, i.e., the markers, are attached directly to the quadrotor UAV, where we set p as the number of markers and denote markers translation in quadrotor body frame ($\mathcal{B}$) as $\mathcal{F} = \{f_1, f_2, \dots, f_p\}$. On the other hand, a set of detection candidates on 2-D image, denoted by $\mathcal{C} = \{c_1, c_2, \dots, c_q\}$, is retrieved. As we will be using the perspective-n-point (PnP) algorithm, p and q should be at least 4, and the correspondence pairs $\langle f_i, c_j \rangle$ should be determined.

When each 2-D image is received, we conduct the detection of $\mathcal{F}$ using conventional image processing methods. In this implementation, as we are using visible LED probe lights, the first objective would be extracting the target light blobs within a noisy image. First, with depth masking τ_{dp} and gray scaling τ_{gy}, a thresholding function suppresses the unwanted pixels to 0

$$\rho'(u, v) = \begin{cases} \rho(u, v), & \text{if } \rho(u, v)_{\text{dp}} < \tau_{\text{dp}} \wedge \rho(u, v)_{\text{gy}} > \tau_{\text{gy}} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where τ_{dp} is the effective landing distance, whereas τ_{gy} depends on the camera's shutter speed. Consequently, the return image is smoothed with Gaussian smoothing, whereas blobs are afterward returned by blobs detection. The 2-D coordinates of extracted blobs are then inserted into candidate set $\mathcal{C}$.

B. Initialization

With $\mathcal{F}$ and $\mathcal{C}$, the correspondences could then be searched. Nevertheless, at $k = 0$, without any prior information, one can only get the matching relation via a brute-force method. Furthermore, to ensure no outlier detection at the start, a stringent initialization condition is set, where we utilize the LED constellation configuration $\mathcal{F}$, the retrieved candidates $\mathcal{C}$, and depth information.

To reject outliers at initialization, the raw 3-D information of $\mathcal{C}$ in the camera frame is first acquired with depth information (s) and the intrinsic model ($\mathbf{K}$) of the camera

$$\boldsymbol{\zeta}_i = \mathbf{K}^{-1} s_i * [u_i, v_i, 1]^T \quad \forall \langle u_i, v_i\rangle \in \mathcal{C}. \tag{6}$$

All $\boldsymbol{\zeta}_i$ then forms set $\mathcal{PCD}$. Successively, we compare the mean average deviation (MAD) of $\mathcal{F}$ and $\mathcal{PCD}$

$$\Delta_{\mathrm{MAD}} = \mathrm{MAD}(\mathcal{F}) - \mathrm{MAD}(\mathcal{PCD}). \tag{7}$$

If Δ_{MAD} is below threshold λ_{MAD}, the program continues to the next step. Otherwise, it declines initialization. In addition, to proceed with initialization, $q = |\mathcal{C}|$ should be equal to the size of $p = |\mathcal{F}|$.

Thereupon, a brute-force search method is used, where raw poses of each correspondence permutation are calculated; the correspondence with the lowest reprojection error is then selected. We first form permutations of $\mathcal{C}$, which is denoted by $\mathcal{Y}$. Meanwhile, as visible LED lights are utilized, additional information from the hue, saturation, and value (HSV) space can reduce the search size. Then, for each permutation ($\mathcal{Y}_j$), a pose $\mathbf{x}_j$ is solved with the PnP algorithm proposed by Lepetit et al. [25]. The reprojection error ($\varepsilon_{\mathcal{Y}_j}$) is then calculated with

$$\varepsilon_{\mathcal{Y}_j} = \sum_{i=1}^{p} \|\mathbf{K}\mathbf{T}(\mathbf{x}_j)\mathcal{F}_i - \mathcal{C}_i\|_2{}^2 \tag{8}$$

where $\mathbf{K}$ is, again, the intrinsic model of the camera. $\mathbf{T}(\cdot)$ is a mapping function that maps $\mathbf{x}$ to $\mathbb{SE}(3)$. The order of $\mathcal{C}_i$ depends on $\mathcal{Y}_j$. $\varepsilon_{\mathcal{Y}_j}$ with the minimum value implies the correct correspondence pair $\mathcal{P}$, which is the final return of the initialization module. Algorithm 1 summarizes the initialization process.

C. Correspondence Tracking

After initialization, to preserve stable PnP measurement, the successive correspondences at $t > 0$ should be retrieved correctly. At each time step k ($k \neq 0$), based on the estimated pose of $\mathbf{x}_{k-1}$ at time $k-1$, we reproject the points in $\mathcal{F}$ onto the image frame, which are denoted by $\mathcal{F}_{2\mathrm{D}}$. Based on the maximum/minimum 2-D values of $\mathcal{F}_{\mathbf{2D}}$, a bounding box **ROI** is generated. With the image filtered by **ROI** mask and depth information $\mathbf{s}_k$, we repeat (5) to generate $\mathcal{C}_k$.

Algorithm 1 Initialization

```
Function Init($\mathcal{F}$, $\mathcal{C}$, hsvimg):
    if sizeof($F$) $\neq$ sizeof($C$) then
        $\mathcal{P}$ = null
        return $\langle$false, $\mathcal{P}\rangle$
    end

    $\mathcal{PCD} \leftarrow$ getPCD($\mathcal{C}$)
    $\Delta_{\mathbf{MAD}} \leftarrow$ getMAD($\mathcal{F}$) $-$ getMAD($\mathcal{MAD}$)
    if $\Delta_{\mathbf{MAD}} > \lambda_{\mathbf{MAD}}$ then
        $\mathcal{P}$ = null
        return $\langle$false, $\mathcal{P}\rangle$
    end

    $\mathcal{Y} \leftarrow$ GenPermute($\mathcal{C}$)
    $\varepsilon' \leftarrow$ inf
    for $\mathcal{Y}_j$ in $\mathcal{Y}$ do
        $\mathbf{hsv}_c \leftarrow$ HSV($\mathcal{Y}_j$, hsvimg)
        if isNeighborHue( $\mathbf{hsv}_f$, $\mathbf{hsv}_c$ ) then
            $\mathbf{x}_j \leftarrow$ EPnP($\mathcal{F}$, $\mathcal{Y}_j$)
            $\varepsilon_{\mathcal{Y}_j} \leftarrow$ getReprojError( $\mathbf{x}_m$, $\mathcal{F}$, $\mathcal{Y}_j$)
            if $\varepsilon_{\mathcal{Y}_j} < \varepsilon'$ then
                $\mathcal{P} \leftarrow \langle\mathcal{F}, \mathcal{Y}_j\rangle$
                $\varepsilon' \leftarrow \varepsilon_{\mathcal{Y}_j}$
            end
        else
            continue
        end
    end
    return $\langle$true, $\mathcal{P}\rangle$
end
```

As $\mathcal{F}_{\mathbf{2D}}$ is directly generated from $\mathcal{F}$, correspondence can be directly retrieved by conducting nearest neighbor with $\mathcal{F}_{\mathbf{2D}}$ and $\mathcal{C}_k$. Nevertheless, in the proposed system, as aggressive maneuvers can occur during landing, the displacement of LED blobs on a 2-D image could be large. Therefore, performing the nearest neighbor search directly with $\mathcal{F}_{2\mathrm{D}}$ derived from pose from $k-1$ could lead to wrong results. In [24], a linear velocity model is used to predict the coordinates of new 2-D blobs. However, such a method might not be stable during large acceleration or when the cluster of blobs is rather small on the image frame due to the relatively large distance between the camera and the quadrotor. Hence, a trivial but efficient modification is applied, in which $\mathcal{F}_{2\mathrm{D}}$ is shifted by the center difference of $\mathcal{F}_{2\mathrm{D}}$ and $\mathcal{C}_k$. We denote the shifted blob coordinates as $\mathcal{F}'_{2\mathrm{D}}$. The nearest neighbor search is then conducted based on $\mathcal{F}'_{2\mathrm{D}}$ and $\mathcal{C}_k$; the $\mathcal{P}$, along with the detected candidates $\mathcal{C}_k$, is then the final measurement output of correspondence tracking.

D. Recursive Filtering—Predict

A recursive filtering method is proposed to track the estimation of the relative states, where an IEKF is adopted. As the state is expressed on manifold, as shown in (1), we involve Lie algebra to perform exponential and logarithm mapping when deriving derivatives and integrals.

In the absence of inertial measurement unit (IMU) data, the nonlinear prediction model is represented with Gaussian noises as follows:

$$\begin{aligned}\dot{\mathbf{p}} &= \mathbf{v}\\ \dot{\mathbf{R}} &= \mathbf{R}\boldsymbol{\xi}_{1\times}\\ \dot{\mathbf{v}} &= \mathbf{R}(\mathbf{g}_{\mathcal{B}}+\boldsymbol{\xi}_2)+\mathbf{g}_{\mathcal{I}}.\end{aligned} \tag{9}$$

From the above equations, $\mathbf{g}_{\mathcal{B}}$ and $\mathbf{g}_{\mathcal{I}}$ are the gravity vectors in the body frame and the inertial frame, respectively. $\boldsymbol{\xi}_1 \in \mathbb{R}^3$ is a Gaussian vector with zero mean; $\boldsymbol{\xi}_2 = [0, 0, \xi_z]$, where ξ_z follows a zero-mean Gaussian distribution. This makes the acceleration induced by the control input $\mathbf{g}_{\mathcal{B}} + \boldsymbol{\xi}_2$, which is assumed to be close to hover thrust. Therefore, the prediction stage is given as follows:

$$\begin{aligned}\hat{\mathbf{x}}_k^- &= \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{w}) = \hat{\mathbf{x}}_{k-1} \boxplus \delta t \delta \mathbf{x}\\ \mathbf{P}_k^- &= \mathbf{F_k P_{k-1} F_k}^T + \mathbf{Q}\end{aligned} \tag{10}$$

where

$$\mathbf{F}_k = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k-1}}. \tag{11}$$

In which, $\mathbf{f}(\mathbf{x}, \mathbf{w})$ is the nonlinear transformation with Euler's method, and $\delta\mathbf{x}$ is derived based on (9). Note that $\boxplus$ operation is defined as follows:

$$\mathbf{A} \boxplus \Delta = \mathbf{A}\mathbf{Exp}(\Delta), \quad \mathbf{A} \in \mathcal{M}, \ \ \Delta \in \mathbb{R}^9 \tag{12}$$

where $\mathbf{Exp}(\cdot)$ is the exponential mapping that maps the Lie algebra of $\mathcal{M}$, i.e., $\mathfrak{m} = \mathfrak{r} \oplus \mathfrak{so}(3) \oplus \mathfrak{r}$, to $\mathcal{M}$ [defined in (1)]. Hence, Δ is defined within the tangent space ($\mathbb{R}^9$) of the Lie group. Furthermore, in (11), the covariance matrix $\mathbf{P}_k^-$ is defined as the local perturbations around $\mathbf{x}_k \in \mathcal{M}$, whereas the $\mathbf{F}_k$ is the derived Jacobian matrix on the Lie group for covariance propagation.

E. Recursive Filtering—Update

The main difference between EKF and IEKF is how the filter updates the state based on priori and measurement, whereas the latter iteratively performs linearization and updates accordingly. Here, we linearize the pin-hole camera model. The main motivation for using IEKF is to alleviate the linearization errors induced in conventional EKF, determining the state as a maximum a posteriori (MAP) estimate. Particularly, we follow the method of [26], in which the posteriori state is returned by performing Gauss–Newton optimization.

During the update stage, in addition to the measurement for pose inference (Section IV-C), we also retrieve velocity measurement based on a constant-velocity model by setting $\mathbf{v}_{\mathbf{z}k}$ as $(\mathbf{p}_{k-1} - \mathbf{p}_{k-2})/\delta t_k$. Nevertheless, as it is derived directly from numerical differentiation, the returned signal is intermingled with high-frequency noise; therefore, we preprocess it with a low-pass infinite impulse response (IIR) filter

$$\mathbf{v}_{\mathbf{z}k} \leftarrow \alpha * \mathbf{v}_{\mathbf{z}k} + (1-\alpha) * \mathbf{v}_{\mathbf{z}k-1} \tag{13}$$

where $\alpha \in [0, 1]$. Subsequently, the update step could be then summarized as in (14). Note that time step k is abbreviated here for readability

$$\hat{\mathbf{x}}_k = \arg\min_{\mathbf{x}} \ \{r_{\mathcal{D}}(\mathbf{x}, \mathbf{x}^-, \mathbf{P}^-) + r_{\mathcal{V}}(\mathbf{x}, \mathcal{C}, \mathbf{v}_{\mathbf{z}}, \mathbf{R})\} \tag{14}$$

$$r_{\mathcal{D}}(\cdot) = \|\mathbf{x} \boxminus \mathbf{x}^-\|^2_{\mathbf{P}^-} \tag{15}$$

$$r_{\mathcal{V}}(\cdot) = \|\mathbf{V}(\mathbf{x}) - \mathbf{v}\|^2_{\mathbf{R}_v} + \sum_{i=0}^{q} \|\mathbf{KT}(\mathbf{x})\mathcal{F}_i - \mathcal{C}_i\|^2_{\mathbf{R}_p}. \tag{16}$$

The nonlinear least squared function consists of two residual functions, $r_{\mathcal{D}}$ and $r_{\mathcal{V}}$, which are, respectively, the dynamic factor and visual factor for the single-time MAP problem. Moreover, $\mathbf{P}^-$ is the priori covariance matrix, and $\mathbf{R}$ is the measurement noise matrix. As for $\mathbf{V}(\cdot)$, it is the mapping function that maps $\mathbf{x}$ to $\mathbb{R}^3$, which is the velocity vector space. As mentioned, optimal $\mathbf{x}_k$ is solved via Gauss–Newton optimization. Also note that $\boxminus$ operation is defined as follows:

$$\mathbf{A} \boxminus \mathbf{B} = \mathbf{Log}(\mathbf{B}^{-1} \circ \mathbf{A}) = \Delta, \quad \mathbf{A}, \mathbf{B} \in \mathcal{M}, \ \ \Delta \in \mathbb{R}^9 \tag{17}$$

where the $\mathbf{Log}(\cdot)$ is the logarithm mapping, inverse mapping of $\mathbf{Exp}(\cdot)$. As for $\circ$, it is the group composition operation within the Lie group.

Afterward, we also propagate the covariance information after all iterations by calculating the Kalman gain based on the final $\hat{\mathbf{x}}_k$

$$\begin{aligned}\mathbf{H}_k &= \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_k}\\ \mathbf{S}_k &= \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}\\ \mathbf{K}_k &= \mathbf{P}_k^- \mathbf{H}_k^T \mathbf{S}_k^{-1}\\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\mathbf{R}\mathbf{K}_k^T.\end{aligned} \tag{18}$$

Particularly, the function $\mathbf{h}$ is the nonlinear transformation between state space and measurement state space

$$\mathbf{z} = \left[\mathcal{C}_k^T, \mathbf{v}_{\mathbf{z}k}^T\right]^T = \mathbf{h}(\mathbf{x}_k). \tag{19}$$

Note that for final covariance propagation, we adopt the Joseph formulation [27] for numerical stability so that matrix symmetry can be ensured. The final retrieved state $\mathbf{x}_k$ is then forwarded as estimation feedback for motion planning.

V. Landing Motion Planning

This section elucidates the motion planning module. To begin with, we program the motion primitive in a relative fashion, meaning that the generated trajectories are coupled with the relative states and relative constraints of the quadrotor with respect to the local frame ($\mathcal{N}$) of the dynamic platform. Furthermore, to ensure the quadrotor stays within the FoV at all times, the trajectory generator takes visibility safety into consideration and imposes kinematic constraints accordingly during the optimization stage. In addition, the Bézier basis is harnessed to guarantee the aforementioned constraints, while time allocation for the trajectory is further discussed to improve the optimality of the result.

A. Motion Planning in Relative Local Frame

With the differential flatness of the quadrotor, the trajectory is parameterized with the flat outputs ${}^{\mathcal{N}}\boldsymbol{\sigma} = [{}^{\mathcal{N}}x, {}^{\mathcal{N}}y, {}^{\mathcal{N}}z, {}^{\mathcal{N}}\psi]^T$, reformulating (3) into

$$ {}^{\mathcal{N}}\boldsymbol{\sigma}(t) = ({}^{\mathcal{N}}x(t), {}^{\mathcal{N}}y(t), {}^{\mathcal{N}}z(t), {}^{\mathcal{N}}\psi(t)) \quad \forall t \in [T_0, T]. \tag{20}$$

As the trajectory is generated within the local frame, to ensure the dynamic feasibility, ${}^{\mathcal{N}}\boldsymbol{\sigma}(t)$ in (20) should be confined accordingly by constraints which are expressed locally in $\mathcal{N}$. Therefore, dynamic constraints $\boldsymbol{\beta}_{\mathrm{D}} \in \mathbb{R}^6$, which comprises velocity and acceleration at time t during optimization, could be retrieved by

$$ {}^{\mathcal{N}}\boldsymbol{\beta}_{\mathrm{D}}(t) = \left[\mathbf{R}_{\mathcal{I}}^{\mathcal{N}}(t) \oplus \mathbf{R}_{\mathcal{I}}^{\mathcal{N}}(t)\right][{}^{\mathcal{I}}\boldsymbol{\beta}_{\mathrm{D}} - {}^{\mathcal{I}}\boldsymbol{\beta}_{\mathrm{N}}(t)]$$

where

$$\begin{aligned} \mathbf{R}_{\mathcal{I}}^{\mathcal{N}} &\in \mathbb{SO}(3) \\ {}^{\cdot}\boldsymbol{\beta}_{\cdot} &\in \mathbb{R}^6 \end{aligned} \tag{21}$$

where $\mathbf{R}_{\mathcal{I}}^{\mathcal{N}}$ rotates translational vectors defined in inertial frame $\mathcal{I}$ to noninertial target frame $\mathcal{N}$; in this case, we apply the direct sum of two identical rotation matrices to rotate the dynamic vector ${}^{\cdot}\boldsymbol{\beta}_{\cdot}$. ${}^{\mathcal{I}}\boldsymbol{\beta}_{\mathrm{D}}$ is the dynamic constraints of the quadrotor defined in $\mathcal{I}$, whereas ${}^{\mathcal{I}}\boldsymbol{\beta}_{\mathrm{N}}$ is the velocity and acceleration of the target frame, i.e., the moving platform defined in $\mathcal{I}$. Note that within the velocity component of ${}^{\mathcal{I}}\boldsymbol{\beta}_{\mathrm{N}}$, the contribution due to the rotation of the noninertial frame $\boldsymbol{\Omega} \times {}^{\mathcal{I}}\mathbf{r}$ is considered. $\mathbf{r}$ is the relative position vector of the UAV with respect to UGV expressed in $\mathcal{I}$. By setting the premise of motion planning in the local frame, we further detail the trajectory generation optimization problem.

B. Visibility-Safety Corridor Generation

In the proposed system, to ensure visibility, it is crucial to constrain the quadrotor within the limited FoV of the camera at all times during the landing stage. To achieve this, we utilize convex polyhedra to construct the safe flight corridor denoted as the visual safe flight corridor (VSFC). Note that VSFC is also expressed in local frame $\mathcal{N}$.

1) Upon initialization of the recursive filter tracker (Section III), a landing cubic free space ($\mathcal{S}$) is generated based on the initial state ($\mathbf{x}_0$) and terminate state ($\mathbf{x}_{\mathrm{T}}$), where $\mathbf{x}_{T_0}, \mathbf{x}_{\mathrm{T}} \in \mathcal{S}$. $\mathcal{S}$ also excludes occupancies $\mathcal{O}$, which constitutes the moving landing platform and the surroundings, and hence, $\mathcal{S} \leftarrow \mathcal{S} \setminus \mathcal{O}$.
2) VSFC generator first segment $\mathcal{S}$ into two: *rear* space ($\mathcal{S}_R$) and *touchdown* space ($\mathcal{S}_{\mathrm{T}}$). Specifically, we confine $\mathcal{S}_{\mathrm{T}}$ with a touch down height (h_{T}) and exclude the space above h_{T}. This is to reduce ground-effect at touchdown, where $\mathcal{S}_{\mathrm{T}}$ is deflated to make the trajectory have less vertical maneuver. $\mathcal{S}_R$ and $\mathcal{S}_{\mathrm{T}}$ are then enlarged with dilate coefficient ϵ_d to guarantee the continuity of the entire landing space so that $\mathcal{S}_R \cap \mathcal{S}_{\mathrm{T}} \neq \emptyset$.
3) We then utilize the FoV information of the camera, i.e., $\theta_w \times \theta_h$, along with sensor translation and rotation in the local frame to construct tangent hyperplanes; each hyperplane induces a half space, the set of half spaces is then expressed as follows:
$$\mathcal{S}_C = \left\{\mathbf{s} \middle| \mathbf{a}_i^T \boldsymbol{\sigma} \leq \mathbf{b}_i \quad \forall i = 0, 1, 2, \ldots, l\right\}. \tag{22}$$
4) Therefore, the final VSFC could be expressed as convex polyhedra
$$\mathcal{S}_L = \{\mathcal{S}_{L_i} | \mathcal{S}_{L_i} = \mathcal{S}_i \cap \mathcal{S}_C, i = R, T\}. \tag{23}$$
From (23), convexity is preserved as each polyhedron is the intersection of convex halfspaces. i denotes the two consecutive landing polyhedra, namely, *rear* and *touchdown*. Finally, to guarantee safety, the robot is modeled as a sphere with radius r_q; we then shrink $\mathcal{S}_L$ with ϵ_s, $\mathcal{S}_L \leftarrow \epsilon_s \mathcal{S}_L$. Fig. 3 exemplifies the defined VSFC.

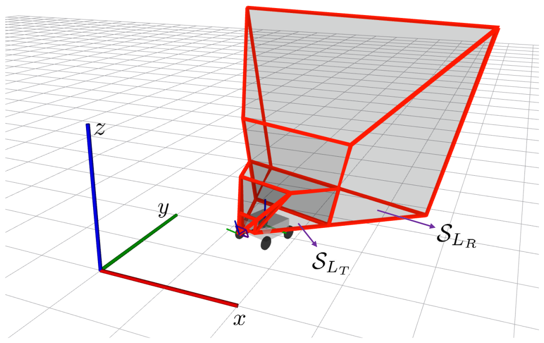


Fig. 3. Graphic presentation of the proposed VSFC $\mathcal{S}_L$, which consists two subsets, *rear* space $\mathcal{S}_{L_R}$ and *touchdown* space $\mathcal{S}_{L_T}$. Visualization package developed by Liu et al. [28] is used here.

C. Trajectory Generation Based on Bézier Basis

A smooth trajectory is then planned as piecewise polynomials, in which flat outputs are defined in (20) are utilized and parameterized with t. Concurrently, the generated trajectory should be subject to constraints defined in Sections V-A and V-B.

A common polynomial expressed with monomial bases is usually structured as follows:

$$\mathbf{p}(t) = \sum_{i=0}^{n} \mathbf{p}_i t^i \tag{24}$$

which is an nth order polynomial. In the proposed system, we modify the conventional polynomial to Bézier curve, a polynomial represented by Bernstein polynomials bases. The main motivation for such reformulation is to ensure the motion primitives stay within the dynamic and kinematic feasible region; this method has been validated by Hönig et al. [29]. First, the original polynomial is recast as follows:

$$\mathbf{p}_{\mathbb{B}}(t) = \sum_{i=0}^{n} \mathbf{c}^i b_n^i(t)$$

where

$$b_n^i(t) = \binom{n}{i} t^i (1-t)^{n-i} \tag{25}$$

where n-degree curve is defined by Bernstein basis $b_n^i(t)$ along with a series of control points $\mathbf{c}^i$. Through utilizing control points, Bézier curve yields two useful properties to allow trajectories to comply with some hard constraints [30].

1) *Convex Hull Property:* Based on the control points $\mathbf{c}^i$, the generated trajectory $\mathbf{p}_{\mathbb{B}}(t)$ is confined within the convex hull constructed by $\mathbf{c}^i$'s. Therefore, by restricting the feasible region of control points based on VSFC (23), the kinematic feasibility could be ensured.
2) *Hodograph Property:* With the hodograph property, the derivatives of the Bézier curve is also a Bézier curve, which could be expressed in terms of another set of $\mathbf{c}^{i'}$, which can formulate another Bézier curve defined in (25). Therefore, by utilizing this characteristic, the dynamic feasibility is guaranteed, where they are confined by (21).

Meanwhile, to generate an optimal trajectory, we minimize the integral of the squared third-derivative (i.e., the jerk) of the trajectory with respect to the weights, i.e., control points $\mathbf{c}^i$, of the trajectory. The objective cost function is shown as follows first:

$$\mathbf{J} = \int_{T_0}^{T} \left\| \frac{d^3\mathbf{p}_{\mathbb{B}}(t)}{dt^3} \right\|^2 dt. \tag{26}$$

We then rewrite the objective in terms of optimization variables($\mathbf{c}$), mapping ($\mathbf{M}$), and differential matrices ($\mathbf{Q}$)

$$\mathbf{J} = \int_{T_0}^{T} \left\| \frac{d^3\mathbf{p}_{\mathbb{B}}(t)}{dt^3} \right\|^2 dt = \mathbf{c}^T\mathbf{M}^T\mathbf{Q}\mathbf{M}\mathbf{c}. \tag{27}$$

As mentioned, we also want the trajectory to comply with feasibilities, and hence, the optimization problem is subject to affine constraints, as stated in (21) and (23), along with equality constraints induced by starting and terminating state, and continuity between each polynomial. The final optimization problem is hence formulated as follows:

$$\begin{aligned} &\min_{\mathbf{c}\in\mathbb{R}^3} \mathbf{c}^T\mathbf{M}^T\mathbf{Q}\mathbf{M}\mathbf{c} \\ &\text{s.t. } \Lambda_{\text{eq}}\mathbf{c} = \boldsymbol{\beta}_{\text{eq}} \\ &\qquad \Lambda_D\mathbf{c} \preceq \boldsymbol{\beta}_D \\ &\qquad \Lambda_S\mathbf{c} \preceq \boldsymbol{\beta}_S \end{aligned} \tag{28}$$

where Λ_{eq} denotes the affine relation matrix between control points $\mathbf{c}$ and equality constraints, while Λ_D and Λ_S map $\mathbf{c}$ to inequality constraints of (21) and (23). From (28), one could easily identify it as a convex problem, as it has a convex objective function and convex constraints. Furthermore, it is also a quadratic programming (QP) problem, which can be easily solved by off-the-shelf QP solvers.

D. On Time Allocation

From (28), it is seen that the derivative matrix $\mathbf{Q}$ is temporally dependent, meaning that $\mathbf{Q}$ changes with different time inputs. Therefore, this implies that with different time allocations, the objective of the optimization problem would differ, leading to suboptimal results. To achieve better results, we sample a sequence of time allocations and evaluate each cost accordingly within the set. Below is the proposed strategy step by step.

1) *Sample on Start:* Once the guidance planner is initialized, a time allocation $\mathcal{T}$ set is constructed, in which each element is sampled within the region of minimum and maximum time calculated based on the velocity in dynamic constraints, namely, the best and worst case scenarios.
2) *Constraints Tightening:* From (28) and (21), it could be readily identified that solving the trajectory online is a perturbed problem [31, p. 249]. Therefore, it is natural to tighten the constraints when optimizing throughout the set so a feasible solution exists during online optimization. Consequently, for (21) during sampling stage, minimum $\boldsymbol{\beta}_{\mathcal{N}}$ and maximum $\boldsymbol{\beta}_{\mathcal{P}}$ are selected to retrieve the most conservative dynamic constraints.
3) *Optimization Throughout the Set:* Each sample time is then visited, and its respective optimization is set up and solved. The problem with the lowest returned cost will yield the best trajectory segment times. The planner solver will then take the optimal time allocation to formulate the objective function.
4) *Online Solving:* As mentioned, the returned time allocation will serve as the optimal segment times. At the landing stage, the planner will use this to construct the $\mathbf{Q}$ matrix accordingly and solve the objective as shown in (28).

E. Feedback Control

Finally, a trajectory parameterized with time t is solved during the landing stage. Based on the polynomial, a series of setpoints could be calculated; with each setpoint at time step t_k and state $\mathbf{x}_k$, a proportional–integral–derivative (PID) + feedforward (FF) controller will then determine the corresponding outer loop control command. Note that the FF term is acquired based on the velocity of the noninertial frame (${}^{\mathcal{I}}\mathbf{v}_{\mathcal{N}}$) and the trajectory velocity expressed in the noninertial frame (${}^{\mathcal{N}}\dot{\boldsymbol{\sigma}}(t)$). Expressing in the inertial frame ($\mathcal{I}$), the FF term is set as follows:

$${}^{\mathcal{I}}\mathbf{u}_{\text{ff}} = \mathbf{K}_{\text{ff}}\big({}^{\mathcal{I}}\mathbf{v}_{\mathcal{N}} + \mathbf{R}^{\mathcal{I}}_{\mathcal{N}}\,{}^{\mathcal{N}}\dot{\boldsymbol{\sigma}}(t)\big). \tag{29}$$

where $\mathbf{K}_{\text{ff}}$ represents the FF gain and $\mathbf{R}$ denotes the rotation matrix. The inclusion of ${}^{\mathcal{I}}\mathbf{v}_{\mathcal{N}}$ serves to compensate for the relative velocity between the two agents in the inertial frame. Subsequently, the onboard attitude controller will track the control signal, which is relayed from the ground station to the quadrotor.

VI. Experimental Results and Discussion

A. Experimental Setup

Experiments presented in this section are done in a controlled environment installed with a motion capture system (VICON); the overall landing mission is guided with a predefined state machine, as shown in Fig. 4. Fig. 5, on the other hand, depicts the experimental setup.

In the proposed work, we validate the design with a minimalistic setup. Specifically, for the quadrotor, we tailor-cut a

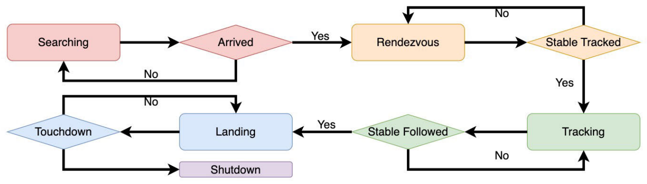


Fig. 4. Predefined finite state machine.

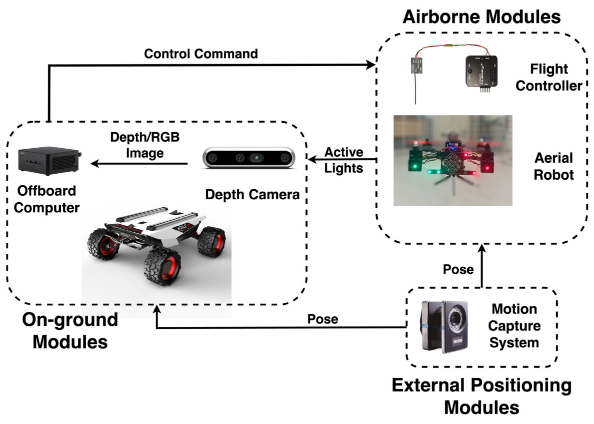


Fig. 5. Experimental setup for the proposed system.

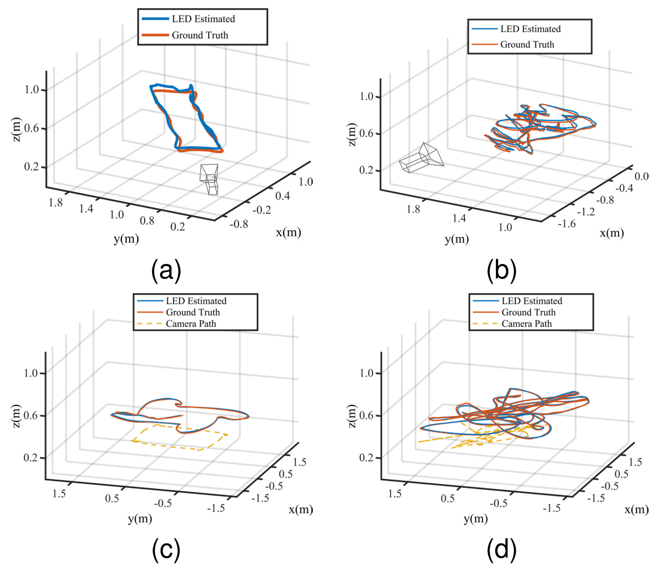


Fig. 6. RSE. Note that the yellow dotted trajectories in Fig. 6(c) and (d) are the movements of the ground camera. (a) Quadrotor moving in a block path in front of the camera. (b) Random maneuver in front of the camera. (c) Camera moving in block path, while the quadrotor follows. (d) Camera moving in the random path, while quadrotor flies with the random path defined in the noninertial frame $\mathcal{N}$.

carbon fiber quadrotor frame with a dimension of 18 × 18 cm and installed an mRo Pixracer controller hardware and a Wi-Fi-equipped MCU module on it. The controller hardware, with the PX4[1] firmware embedded, provides the inner loop control, whereas the Wi-Fi module is used to receive outer loop control signals from the ground. In addition to the above, we install the LED markers at the front of the quadrotor for the pose estimation module, which is mounted with a configuration that avoids symmetry, co-linearity, self-similarity, and co-planarity, preventing degenerated measurement. In addition, we set the number of LEDs `LED_no` to be larger than 4, where we installed six LED markers to provide redundancy.

Meanwhile, for on-ground modules, AgileX Scout Mini[2] is opted for the main platform, on which we place the landing pad (43 × 43 cm), depth camera (Intel RealSense D455), and a computer (Intel NUC minicomputer, NUC8i7BEH); we fixate the camera with an upward-facing 20 dg. The camera features an FoV of 87 × 58 dg, with an ideal depth range extending up to 6 m and an accuracy of less than 2% at 4 m. It is equipped with a global shutter for RGB image capture, and the image resolution is set to 848 × 640 pixels at a frame rate of 60 Hz.

B. Relative State Estimation

The RSE is assessed by the following: 1) direct evaluation with ground truth data when the camera is still; 2) direct evaluation with ground-truth data when the camera is in motion; and 3) comparison with other RSE methods. Note that the ground-truth data ($\bar{\mathbf{x}}$) is obtained from VICON, which is reported to have an accuracy of less than 0.5 mm. Based on the collected data, we then calculate the absolute trajectory error (ATE) as the evaluation metric

$$e_{\text{ATE}} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{1}{K}\sum_{k=1}^{K}\|\mathbf{x}_{k,i} \boxminus \bar{\mathbf{x}}_{k,i}\|_2^2}. \tag{30}$$

The operator $\boxminus$ is defined similar to 17, but the output is within the tangent space of the Lie group $\Delta \in \mathbb{R}^6$. In addition, the tracking success rate is determined by evaluating the reprojection error for each frame. A frame is considered successfully tracked if its total reprojection error is less than $e <$ `led_no` $\times\, 2$. The success rate is then calculated as the percentage of frames that meet this criterion.

1) Camera at Still: Fig. 6(a) and (b) first showcase the recorded estimation and ground-truth trajectories when the camera is fixed within the inertial frame $\mathcal{I}$. Specifically, in Fig. 6(a), we programmed a rectangular path within the noninertial frame $\mathcal{N}$ for the UAV to follow. As for Fig. 6(b), we randomly controlled the UAV in front of the camera. From Table I, it can be seen that the designed system yields satisfactory results, where the position ATE is less than 0.03 m, whereas the orientation ATE is smaller than 4 dg in the static case. Furthermore, the tracking success rate is 100%.

2) Camera at Motion: Datasets with the camera in motion were also collected, as shown in Fig. 6(c) and (d). Specifically, in Fig. 6(c), we programmed the UGV to move in a block path and let the quadrotor fix at a relative position defined in noninertial frame $\mathcal{N}$; in Fig. 6(d), UGV was controlled manually, and UAV was designed to move in a block path in $\mathcal{N}$. Table I shows that the overall accuracy is within acceptable range for both translation and rotation, with position ATE less than 0.05 m and the orientation ATE less than 5 dg. In addition, the tracking success rate is 99.78%. Therefore, we deem the overall performance stable, whether the camera is static or dynamic. The module could, hence, provide reliable state estimation feedback for our nonrobocentric landing framework.

[1]https://px4.io/
[2]https://global.agilex.ai/products/scout-mini

TABLE I
RESULTS FROM RSE

	Static	Dynamic	Unit
ATE_translation	0.021	0.049	m
ATE_rotation	3.501	4.727	dg
Success Rate	100.00	99.78	%

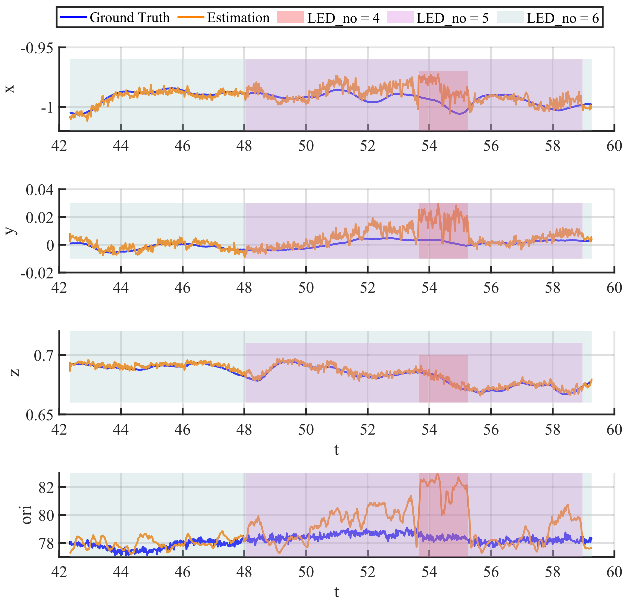


Fig. 7. Estimation results when the LEDs are partially occluded.

From the experiments, the relative distance between the UAV and the ground camera ranges from 0.4 to 2.9 m, with the program consistently achieving a high tracking success rate. Overall, these results demonstrate the robustness and effectiveness of our estimation module.

3) Occlusion Analysis: We also analyze how the estimation results are affected when some of the LEDs are occluded. The experiment involves hovering the UAV in front of the camera and occluding 1–2 LEDs, as the algorithm requires at least four LEDs for accurate estimation. Fig. 7 illustrates the estimation results, with tracking for `LED_no` values of 4–6 represented by red, purple, and green regions, respectively. The orientation metric is presented with axis-angle representation. The results indicate that as the number of LEDs decreases, the estimation error increases, particularly for rotation estimation. However, the overall estimation remains within acceptable limits.

This demonstrates that the estimation module remains functional even in the presence of occlusion. In real-world implementations, a fail-safe mechanism or redundancy system can be employed to mitigate such issues. In addition, using infrared spectrum LEDs can help reduce interference from external visible light and enhance robustness.

4) Comparison With Others: We also compare the proposed IEKF method with other methods: iterative method [24] (denoted by KF-less) and ArUco. The ArUco tag is fixated on the drone, similar to the attached constellation. The comparison is carried out on several datasets with two scenarios: 1) quadrotor following UGV in a circle path and 2) quadrotor landing on a moving platform. Table II reveals the performances of each method. From the table, it could be seen that IEKF returns better accuracy in both scenarios in terms of translation and rotation, where IEKF gives an average of 0.036- and 0.053-m position error as well as 3.104 dg and 3.007 dg orientation error. Furthermore, the tracking success rate is also superior to previous methods, as IEKF successfully tracks the pose at all times, whereas the KF-less method loses a few frames and ArUco gives unstable tracking results. Also, as ArUco cannot track most frames in the dynamic landing dataset, results for that case in this scenario are omitted here. Based on these results, we can then proceed with landing experiments.

TABLE II
RESULTS FROM COMPARISON

		IEKF	KF-less [24]	ArUco [32]
Circle	ATE_translation	0.036 m	0.059 m	0.149 m
	ATE_rotation	3.104 dg	5.497 dg	41.77 dg
	Success Rate	100 %	99.7 %	28.2 %
Dyn. Land	ATE_translation	0.053 m	0.083 m	N/A
	ATE_rotation	3.007 dg	8.956 dg	N/A
	Success Rate	100 %	99.5 %	N/A

TABLE III
LANDING SEQUENCES AND PARAMETERS SETTING

Sequence	Scene	Radius	ang_rate	linear_vel.
M_A	Circular	1.4 m	12 dg/s	N/A
M_B	Circular	1.4 m	24 dg/s	N/A
M_C	Circular	1.4 m	36 dg/s	N/A
M_D	Circular	1.4 m	48 dg/s	N/A
M_E1	Circular (varied)	1.4 m	$12 - 48$ dg/s	N/A
M_E2	Circular (varied)	1.4 m	$12 - 48$ dg/s	N/A
M_F	Spinning	0.0 m	48 dg/s	N/A
M_L	Linear	N/A	N/A	0.6 m/s
M_M1	Manual (random)	N/A	N/A	N/A
M_M2	Manual (random)	N/A	N/A	N/A

C. Landing Experiment

Based on the acquired state data, this section presents the landing mission.

1) Landing Sequences: To assess the robustness of the proposed system, we design a variety of UGV motions, as detailed in Table III. Specifically, sequences M_A through M_D involve uniform circular motions with angular rates varying from 12 to 48 dg/s. Sequences M_E1 and M_E2 feature nonuniform circular motions with varying angular rates. For M_E1, the angular rate varies periodically in the sequence 12, 24, 36, 48, 12 dg/s, ..., with a time interval of 1 s. Sequence M_E2 exhibits a repeating angular rate pattern of 12, 24, 36, 48, 36, 24 dg/s, ..., also with a 1-s time interval. In addition, sequence M_F involves only spinning motion, while M_L represents linear motion. Finally, M_M1 and M_M2 consist of random motions controlled manually.

Table IV presents the results, including the estimation error, trajectory tracking error, and both the average and maximum velocities of the UAV and UGV, as well as the angular rate of the landing platform during the landing phase. Note that the orientation metric here is expressed within

TABLE IV
PERFORMANCE RESULT FROM DIFFERENCE LANDING SEQUENCES

metric	est_err. trans.	est_err. rot.	traj_err. x	traj_err. y	traj_err. z	UAV_vel. avg/max	UGV_vel. avg/max	UGV_vel. avg_r_rate
unit	m	dg	m	m	m	m/s	m/s	dg/s
M_A	0.014	2.370	0.036	0.071	0.043	0.38/1.04	0.29/0.35	11.18
M_B	0.019	2.350	0.051	0.073	0.038	0.79/1.39	0.62/0.66	22.33
M_C	0.025	2.702	0.104	0.059	0.030	1.23/1.98	0.96/1.03	33.52
M_D	0.026	3.048	0.127	0.118	0.036	1.69/2.54	1.34/1.45	44.72
M_E1	0.021	4.181	0.110	0.047	0.031	0.88/1.92	0.68/1.10	24.63
M_E2	0.022	2.862	0.111	0.053	0.054	0.99/1.63	0.77/1.31	27.93
M_F	0.015	2.750	0.120	0.117	0.038	0.54/1.58	0.09/0.43	45.63
M_L	0.016	2.860	0.077	0.063	0.041	0.57/1.80	0.4/1.13	1.39
M_M1	0.015	1.947	0.086	0.048	0.042	0.65/1.41	0.49/1.43	8.52
M_M2	0.013	2.922	0.050	0.119	0.050	0.44/1.24	0.27/0.50	10.70

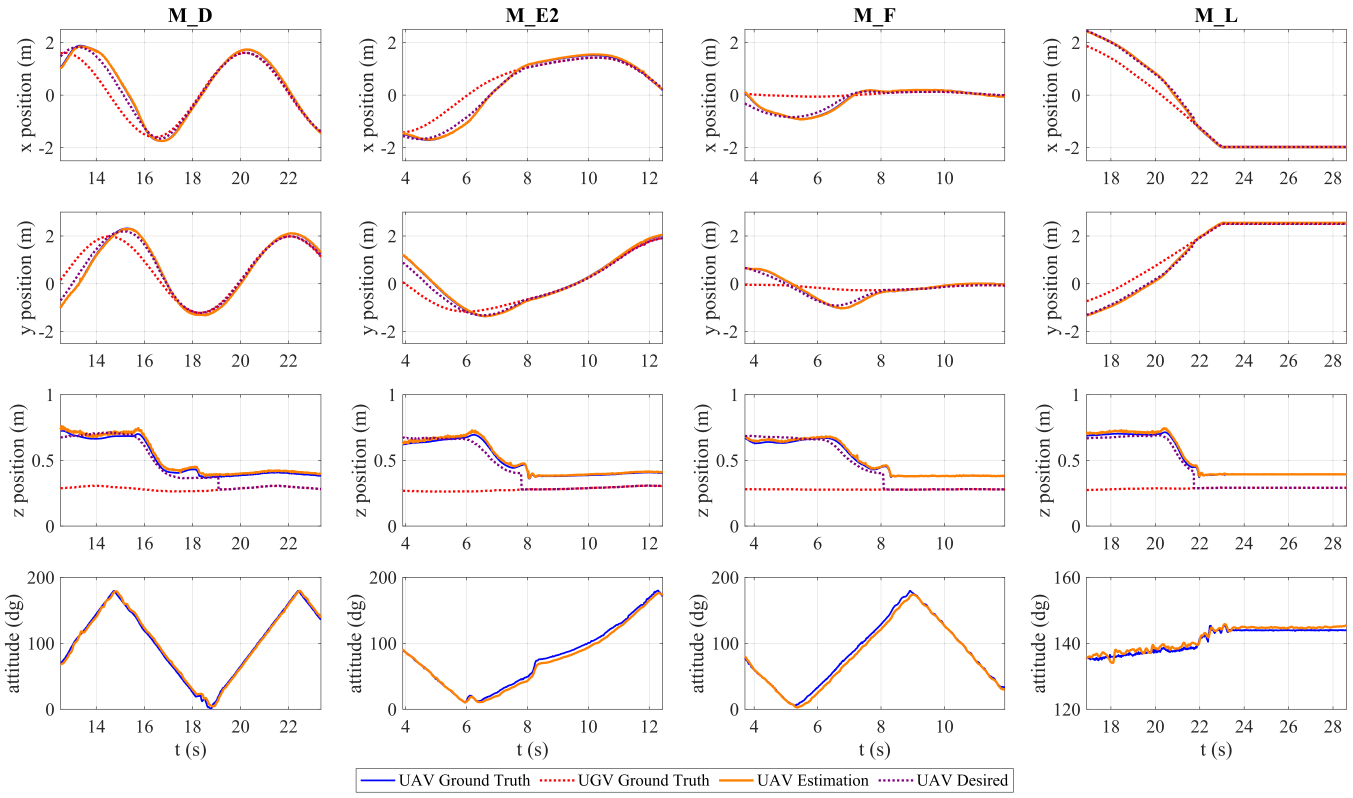


Fig. 8. Pose information of UAV ground truth, UGV ground truth, UAV estimation, and UAV desired (setpoint). From right to left, landing sequence M_D, M_E2, M_F, and M_L are presented, where the details are denoted in Table IV.

the axis-angle representation. The table indicates that more aggressive landing platform motions are associated with larger estimation and tracking errors. However, all results remain within acceptable limits. In addition, landing sequences with higher linear and angular velocities require the UAV to operate at higher velocities as well.

We plot the position, attitude, and velocity information for sequences M_D, M_E2, M_F, and M_L to provide a clearer visualization, as shown in Figs. 8 and 9. Fig. 8 demonstrates that the system consistently achieves stable estimation results for position and rotation during landing across all sequences. Even without onboard exteroceptive sensors and computers, the quadrotor successfully performs landings on a constantly moving landing pad. In addition, the trajectory tracking performance is satisfactory, as indicated by the results. Fig. 9 displays the velocity information from the landing experiments. Although velocity estimation shows relatively larger errors and fluctuations compared to pose estimation, the results are deemed sufficiently accurate for this application, given the absence of IMU data.

2) Comparison With the Absence of Landing Frame Velocity Information: As previously mentioned, the tracking controller utilizes the velocity information of the landing platform and trajectory. To evaluate the impact of ground

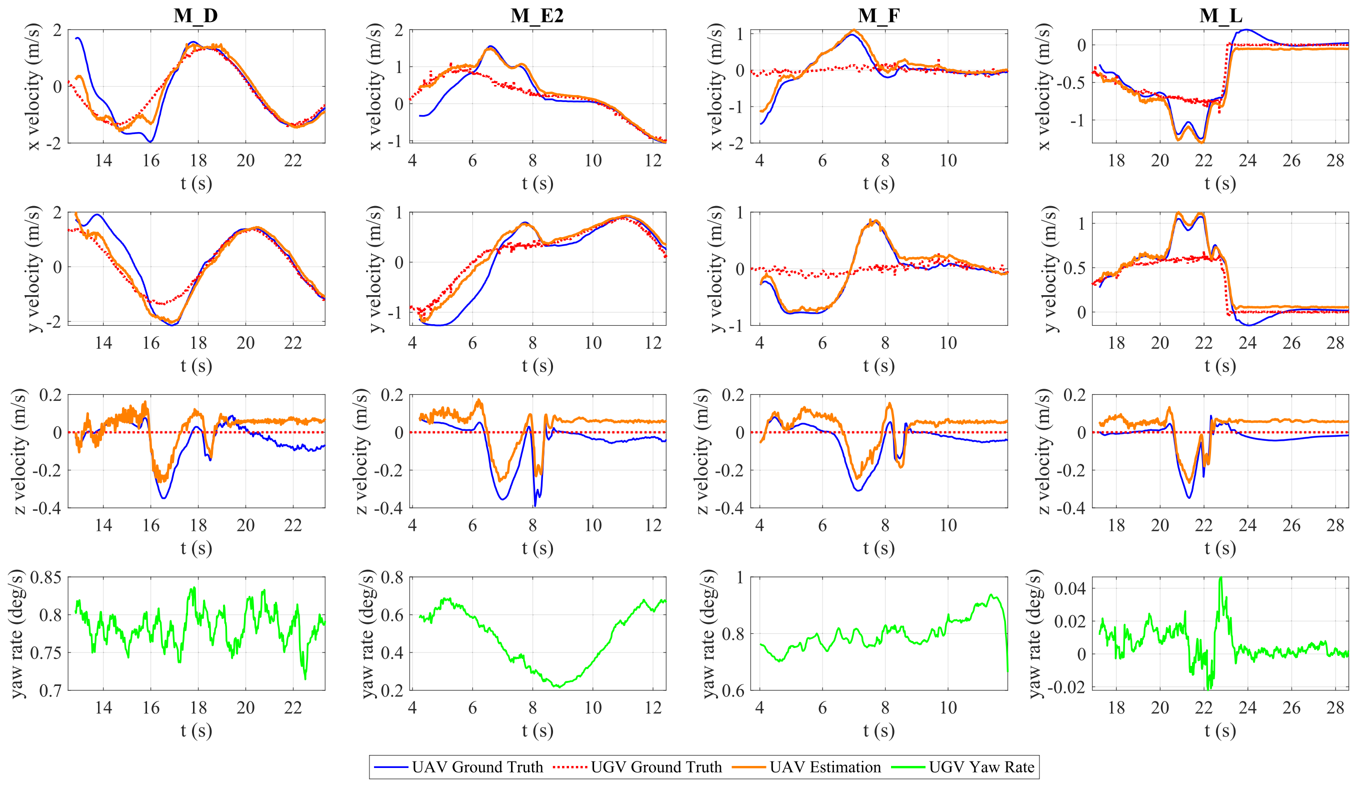


Fig. 9. Velocity information of UAV ground truth, UGV ground truth, UAV estimation, and UAV desired (setpoint). From right to left, landing sequence M_D, M_E2, M_F, and M_L are presented, where the details are denoted in Table IV.

TABLE V
LANDING METHODS COMPARISON (UNIT: m)

		Proposed	without vel_info
M_B	RMSE	0.096	0.211
	err_std	0.044	0.126
	err_max	0.175	0.650
M_L	RMSE	0.108	0.165
	err_std	0.032	0.071
	err_max	0.147	0.324

velocity information, we compared systems with and without complete landing target velocity data (e.g., the airborne system in [5, p. 206]). This comparison was conducted using landing sequences M_B and M_L, which are achievable with different methods. The results, shown in Table V, indicate that with full velocity information, the tracking error for the same trajectory can be reduced, leading to more stable landing performance in certain scenarios.

3) Affects on Landing Pad Size: In addition, we analyze the relationship between landing pad size and the landing success rate. We conducted an independent experimental session with landing sequences M_A–M_F performed 111 times (with each sequence occurring an evenly distributed number of times with random bias). We plot the touchdown positions in the noninertial frame, as shown in Fig. 10. Using a 43 × 43 cm landing pad, we segment the area into three regions: 33 × 33, 23 × 23, and 13 × 13 cm. These regions correspond to effective landing areas for landing pads of sizes 43 × 43, 33 × 33, and 23 × 23 cm, respectively. Based on the

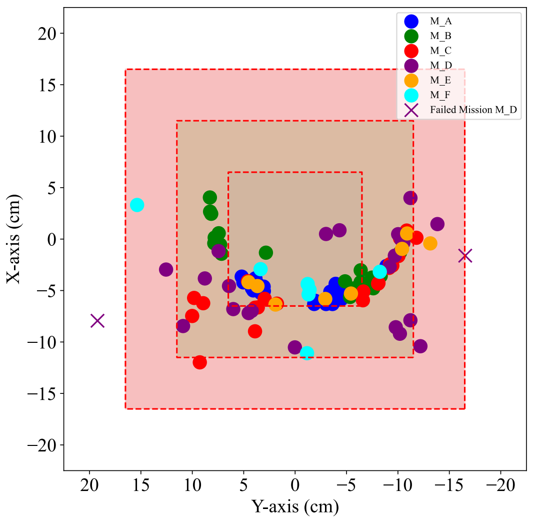


Fig. 10. Scattered dots of touchdown position within the $\mathcal{N}$ frame. The cross denotes the missions that failed.

touchdown positions, we calculate the landing success rate, as presented in Table VI. From the table, it can be concluded that with a landing pad size 33 × 33 cm or larger, most landing sequences can have a relatively high success rate. From sequence M_C and M_D, it can also be inferred that a consistent change in the landing platform's heading angle can lead to a significant drop in success rate, requiring a larger pad size for touchdown guarantee.

Overall, the above results suggest that with this landing configuration, satisfactory landing performance can be achieved.

TABLE VI
TOUCHDOWN SUCCESS RATE

Mission Sequences	Landing Pad Size	Success Rate
M_A	23 × 23 cm	91.67 %
	33 × 33 cm	100.0 %
	43 × 43 cm	100.0 %
M_B	23 × 23 cm	33.33 %
	33 × 33 cm	100.0 %
	43 × 43 cm	100.0 %
M_C	23 × 23 cm	12.50 %
	33 × 33 cm	91.67 %
	43 × 43 cm	100.0 %
M_D	23 × 23 cm	12.50 %
	33 × 33 cm	79.17 %
	43 × 43 cm	91.67 %
M_E	23 × 23 cm	62.50 %
	33 × 33 cm	87.50 %
	43 × 43 cm	100.0 %
M_F	23 × 23 cm	57.14 %
	33 × 33 cm	85.71 %
	43 × 43 cm	100.0 %

TABLE VII
COMPUTATIONAL COSTS (UNIT: ms)

		mean	std.	max
RSE	Initialization	30.56	6.81	42.23
	Image Processing	3.05	0.79	13.95
	Correspondence Search	0.22	0.10	0.70
	IEKF	4.17	1.18	17.28
	Online Total	7.44	1.83	31.93
Planner	Time Allocation Search	659.30	17.59	700.00
	Online Optimization	2.26	2.03	4.00
	Controller	<0.01	<0.01	<0.01
	Online Total	2.26	2.03	4.00

This demonstrates a promising alternative for dynamic landing missions.

D. Computational Load

Finally, we analyze the computational load of the system. In the proposed landing configuration, all modules are operated on the ground. This setup facilitates more powerful computation since ground vehicles can house larger computers. Despite this, the adopted algorithms still enable real-time applicability. Furthermore, all packages are programmed in C++ /Python within the framework of ROS. As for QP solving during motion generation, we use operating splitting quadratic program (OSQP) [33], which can provide efficient solutions within milliseconds and an easy interfacing module for integration.

Specifically, the ground computer operates several threads in parallel, executing the following modules: 1) RSE; 2) trajectory planning; 3) quadrotor outer loop control; and 4) ground vehicle path planning and control.

Table VII presents the statistics on the time demands of the algorithm. The term "RSE" refers to the relative state estimation module, while "Planner" denotes the landing module. The data indicate that the average runtime frequency for the RSE is approximately 130 Hz, whereas the planner's online update rate exceeds 400 Hz. In practice, the camera used provides a frame rate of 60 Hz, and the controller module's input frequency is fixed at 50 Hz.

Although the initial time allocation search for the planning modules requires a significant amount of processing time, this process is executed only once at launch. Since the programs operate in parallel, the impact of this initial processing time is negligible.

VII. CONCLUSION

In this study, we propose a novel quadrotor landing system, termed a "nonrobocentric" (noninertial) framework, designed to operate without onboard exteroceptive sensors and a computer. While we have validated the system in a controlled environment, we posit that this marks a crucial advancement toward an innovative autonomous landing framework, capable of mitigating the heavy reliance on onboard computers and exteroceptive sensors. This, in turn, has the potential to significantly reduce overall costs for the future large-scale deployment of UAV swarms in smart-city scenarios.

In the near future, our research aims to extend the validation of our system by conducting experiments on outdoor moving vehicles and marine vessels, assessing its feasibility for real-world deployment. However, prior to these endeavors, we recognize the necessity for enhancements in the RSE module, particularly focusing on improving range and stability. This aspect is acknowledged as part of our ongoing and future work.

ACKNOWLEDGMENT

Hardware and software are detailed at: https://github.com/HKPolyU-UAV/alan.git

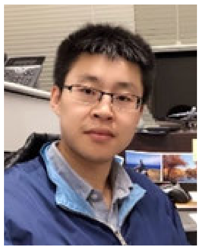
REFERENCES

[1] P. Tokekar, J. V. Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic UAV and UGV system for precision agriculture," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1498–1511, Dec. 2016.
[2] J. Sun, B. Li, Y. Jiang, and C.-Y. Wen, "A camera-based target detection and positioning UAV system for search and rescue (SAR) purposes," *Sensors*, vol. 16, no. 11, p. 1778, 2016.
[3] Q. Zhou et al., "Development of fixed-wing UAV 3D coverage paths for urban air quality profiling," *Sensors*, vol. 22, no. 10, p. 3630, May 2022.
[4] D. Dissanayaka, T. R. Wanasinghe, O. D. Silva, A. Jayasiri, and G. K. I. Mann, "Review of navigation methods for UAV-based parcel delivery," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 1, pp. 1068–1082, Jan. 2024.
[5] D. Falanga, A. Zanchettin, A. Simovic, J. Delmerico, and D. Scaramuzza, "Vision-based autonomous quadrotor landing on a moving platform," in *Proc. IEEE Int. Symp. Saf., Secur. Rescue Robot. (SSRR)*, Oct. 2017, pp. 200–207.
[6] C.-W. Chang et al., "Proactive guidance for accurate uav landing on a dynamic platform: A visual-inertial approach," *Sensors*, vol. 22, no. 1, p. 404, 2022.
[7] D. Lee, T. Ryan, and H. Jin. Kim, "Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 971–976.
[8] L. Coutard, F. Chaumette, and J.-M. Pflimlin, "Automatic landing on aircraft carrier by visual servoing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2011, pp. 2843–2848.
[9] 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 Trans. Robot.*, vol. 32, no. 6, pp. 1524–1535, Dec. 2016.

[10] J. S. Wynn and T. W. McLain, “Visual servoing with feed-forward for precision shipboard landing of an autonomous multirotor,” in *Proc. Amer. Control Conf. (ACC)*, Jul. 2019, pp. 3928–3935.
[11] C. Chen, S. Chen, G. Hu, B. Chen, P. Chen, and K. Su, “An auto-landing strategy based on pan-tilt based visual servoing for unmanned aerial vehicle in GNSS-denied environments,” *Aerosp. Sci. Technol.*, vol. 116, Sep. 2021, Art. no. 106891.
[12] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, “SVO: Semidirect visual odometry for monocular and multicamera systems,” *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, Apr. 2016.
[13] T. Baca et al., “Autonomous landing on a moving vehicle with an unmanned aerial vehicle,” *J. Field Robot.*, vol. 36, no. 5, pp. 874–891, 2019.
[14] S. Yang, S. A. Scherer, and A. Zell, “An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle,” *J. Intell. Robotic Syst.*, vol. 69, nos. 1–4, pp. 499–515, Jan. 2013.
[15] P. Vlantis, P. Marantos, C. P. Bechlioulis, and K. J. Kyriakopoulos, “Quadrotor landing on an inclined platform of a moving ground vehicle,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 2202–2207.
[16] Y. Meng, W. Wang, H. Han, and J. Ban, “A visual/inertial integrated landing guidance method for UAV landing on the ship,” *Aerosp. Sci. Technol.*, vol. 85, pp. 474–480, Feb. 2019.
[17] Y. Qi, J. Jiang, J. Wu, J. Wang, C. Wang, and J. Shan, “Autonomous landing solution of low-cost quadrotor on a moving platform,” *Robot. Auto. Syst.*, vol. 119, pp. 64–76, Sep. 2019.
[18] A. Paris, B. T. Lopez, and J. P. How, “Dynamic landing of an autonomous quadrotor on a moving platform in turbulent wind conditions,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 9577–9583.
[19] C. Martínez, P. Campoy, I. Mondragón, and M. A. Olivares-Méndez, “Trinocular ground system to control UAVs,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2009, pp. 3361–3367.
[20] W. Kong, D. Zhang, X. Wang, Z. Xian, and J. Zhang, “Autonomous landing of an UAV with a ground-based actuated infrared stereo vision system,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 2963–2970.
[21] W. Kong et al., “A ground-based optical system for autonomous landing of a fixed wing UAV,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 4797–4804.
[22] T. Ferreira, A. Bernardino, and B. Damas, “6D UAV pose estimation for ship landing guidance,” in *Proc. OCEANS: San Diego—Porto*, Sep. 2021, pp. 1–10.
[23] J. Sola, J. Deray, and D. Atchuthan, “A micro lie theory for state estimation in robotics,” 2018, *arXiv:1812.01537*.
[24] M. Faessler, E. Mueggler, K. Schwabe, and D. Scaramuzza, “A monocular pose estimation system based on infrared LEDs,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 907–913.
[25] V. Lepetit, F. Moreno-Noguer, and P. Fua, “EP*n*P: An accurate $O(n)$ solution to the P*n*P problem,” *Int. J. Comput. Vis.*, vol. 81, pp. 155–166, Feb. 2009.
[26] B. M. Bell and F. W. Cathey, “The iterated Kalman filter update as a Gauss-Newton method,” *IEEE Trans. Autom. Control*, vol. 38, no. 2, pp. 294–297, Feb. 1993.
[27] R. G. Brown and P. Y. Hwang, *Introduction To Random Signals and Applied Kalman Filtering: With MATLAB Exercises and Solutions*. Hoboken, NJ, USA: Wiley, 1997.
[28] S. Liu et al., “Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments,” *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1688–1695, Jul. 2017.
[29] W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, and N. Ayanian, “Trajectory planning for quadrotor swarms,” *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 856–869, Aug. 2018.
[30] R. F. Riesenfeld, *Applications of B-spline Approximation To Geometric Problems of Computer-aided Design.* Syracuse, NY, USA: Syracuse University, 1973.
[31] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[32] S. Garrido-Jurado, R. Mu noz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, “Automatic generation and detection of highly reliable fiducial markers under occlusion,” *Pattern Recognit.*, vol. 47, no. 6, pp. 2280–2292, Jun. 2014.
[33] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, “OSQP: An operator splitting solver for quadratic programs,” *Math. Program. Comput.*, vol. 12, no. 4, pp. 637–672, Dec. 2020.

Li-Yu Lo (Associate Member, IEEE) received the B.Eng. degree from the Department of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University, Hong Kong, in 2021.

He is currently a member with the AIRo-Lab, Department of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University. His research interests include sensor fusion, multi-robot pose estimation, and heterogeneous robotics systems.


Boyang Li received the B.Eng. and M.Eng. degrees in aeronautical engineering from Northwestern Polytechnical University, Xi’an, China, in 2012 and 2015, respectively, and the Ph.D. degree in mechanical engineering from The Hong Kong Polytechnic University, Hong Kong, in 2019.

He has conducted post-doctoral research at the Air Traffic Management Research Institute, Nanyang Technological University, Singapore, and also with the School of Engineering, The University of Edinburgh, Edinburgh, U.K. In 2020, he established the Autonomous Aerial Systems Laboratory, The Hong Kong Polytechnic University. He joined the University of Newcastle, Callaghan, Newcastle, NSW, Australia, as a Lecturer in aerospace systems engineering in 2023. His research interests include model predictive control, path/trajectory optimization, and field experiments of unmanned aircraft systems (UAS) and other mobile robots.

Chih-Yung Wen received the B.S. degree from the Department of Mechanical Engineering, National Taiwan University, Taipei, Taiwan, in 1986, and the M.S. and Ph.D. degrees from the Department of Aeronautics, California Institute of Technology (Caltech), Pasadena, CA, USA, in 1989 and 1994, respectively.

He joined the Department of Mechanical Engineering, The Hong Kong Polytechnic University, Hong Kong, in 2012, as a Professor. In 2021, he became the Chair Professor and the Head of the Department of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University. His current research interests include modeling and control of tail-sitter unmanned aerial vehicles (UAVs), visual–inertial odometry systems for UAVs, and artificial intelligence (AI) object detection by UAVs.

Ching-Wei Chang received the B.Eng. degree from the Department of Mechanical Engineering, Yuan Ze University, Taoyuan, Taiwan, in 2015, and the Ph.D. degree in mechanical engineering from The Hong Kong Polytechnic University, Hong Kong, in 2022.

He is currently a Researcher with Hong Kong Center for Construction Robotics, The Hong Kong University of Science and Technology, Hong Kong. His research interests include vertical takeoff and landing (VTOL) unmanned aerial vehicles (UAVs), UAV inspection applications, and autonomous landing systems for UAVs.