

Autonomous Drone Racing Via Vision-Only Monocular SLAM and Submapping

L. Oyuki Rojas-Perez¹, Aldrich A. Cabrera-Ponce², Nilda Gabriela Xolo-Tlapanco¹,
Jose Martinez-Carranza^{1,*}

¹ Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Puebla,
Mexico

² Benemerita Universidad Autónoma de Puebla (BUAP),
Mexico

carranza@inaoe.mx

Abstract. We present an Unmanned Aerial System (UAS) for Autonomous Drone Racing (ADR) in which state estimation relies exclusively on monocular Simultaneous localization and Mapping (SLAM). In contrast to approaches that depend heavily on known gate geometry and tightly coupled visual-inertial fusion, we propose a lightweight vision-only solution in which gate information is used solely during map initialisation to recover the metric scale. To mitigate long-term drift, the proposed system employs a submap decomposition strategy that exploits prior knowledge of the race-track layout. For flight control, a cascade PID architecture was implemented to generate roll, pitch, and yaw angle-rate commands transmitted to the flight controller. The complete system operated fully onboard the quadrotor without GPU acceleration at a frequency of 20 Hz, achieving speeds of up to 3 m/s whilst successfully completing the race track autonomously.

Keywords. Autonomous drone racing, visual SLAM, MAV, autonomous flight, UAS, aerial robotics.

1 Introduction

Autonomous Drone Racing (ADR), inspired by human-piloted drone racing, was introduced in 2016 at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). The first teams addressing the challenge demonstrated relatively limited capabilities in terms of flight speed and manoeuvrability [18, 17, 12]. From its inception, the competition imposed strict

operational constraints: all computation had to be performed fully onboard the aerial vehicle, whilst the use of external sensing systems, such as motion-capture infrastructures, was prohibited. Despite the diversity of proposed approaches, three characteristics were common across all participating teams: i) the use of quadrotor platforms; ii) the adoption of cameras as the primary sensing modality for perception and state estimation; and iii) the solution of the top three teams was based on a reactive navigation approach.

Remarkably, for the second edition of the ADR at the 2017 IEEE/RSJ IROS, the 1st- and runner-up teams used a solution based on visual Simultaneous localization and Mapping (SLAM) [17]; the first team won by using a monocular SLAM approach with scale recovery based on the drone's altitude measurement and a planar ground assumption; whereas the second place used an RGB-D SLAM, both approaches used their SLAM estimates for their control strategy.

The ADR challenge attracted the attention of organisations outside of the academic world. In 2019, the company Lockheed Martin organised the AlphaPilot AI Drone Innovation Challenge [10]. The winning team presented a strategy based on state estimation for PID-like controllers [7].

The breakthrough in this challenge came in 2023, when it was published in Nature that the first



Fig. 1. We present an Unmanned Aerial System for Autonomous Drone Racing, which was deployed at the Test Arena of the A2RL drone championship, Season 2. Our approach relies on a metric monocular SLAM with a submapping strategy, which enabled our drone to complete the track autonomously at a maximum flight speed of 3 m/s, finishing the track successfully and even being able to perform 2 laps

autonomous drone to compete at champion-level was able to beat human champions in most of the races, with onboard processing, while combining a stack of techniques to enable a fast, agile flight of 22 m/s [13]. It is worth mentioning that the control technique was based on Reinforcement Learning (RL), however, it relied on state estimation from a visual odometry system corrected with pose estimation from gate observation. Similarly, in 2025 and 2026, the A2RL Autonomous Drone Championship, carried out in Abu Dhabi, gathered some of the best teams in the world to put their methods to the test. In the first season, apart from the champion team [8], which used a technique similar to the ETH group, based on RL, the rest of the teams relied on visual odometry or visual SLAM methods for state estimation coupled with classical control such as the Model Predictive Controller (MPC). In contrast, for the second season in 2026, the top three teams presented solutions based on RL and state estimation based on gate observation.

Although recent trends in ADR suggest an increasing adoption of Reinforcement Learning (RL) techniques over classical control strategies, state estimation remains a fundamental bottleneck for high-speed autonomous flight. In both seasons of the A2RL competition, reliable relative localization based on gate observations proved

critical for successful navigation. As illustrated in Figure 1, corresponding to the test arena used during Season 2, the environment was carefully controlled to facilitate robust gate detection, including the use of homogeneous illumination conditions and visually simplified surroundings with limited scene texture, such as black curtains or textureless floor carpet. Whilst these measures improve the reliability of perception pipelines based on gate-relative localization, they also reduce visual ambiguity in ways that may not generalise to less structured or more realistic deployment scenarios.

In this sense, we argue that, in order to achieve general ADR, the drone should be able to fly agnostic to its environment. This calls for methods such as visual SLAM, which aims at solving the localization problem by building a model of the scene that is being observed while navigating, independently of the presence of certain structures such as gates or any other object. Indeed, the best SLAM systems are still behind in terms of operation frequency (< 30 Hz). While a tightly-coupled visual-inertial odometry solution, such as OpenVINS [9], can achieve a 100 Hz, but with the caveat that IMU readings saturate at high speeds as reported in [29].

Motivated by the above observations and considering the rapid progress of visual SLAM techniques, this work explores the use of a monocular SLAM solution that leverages prior knowledge of the ADR track layout. In the competition setting, the size, position, and orientation of the gates are known beforehand and provided to all participants. Building upon this information, we propose: i) using the known gate dimensions to recover the metric scale during map initialisation, enabling the SLAM system to produce localization estimates directly in world coordinates; ii) exploiting the fact that the track geometry remains fixed, allowing maps to be generated offline and subsequently reloaded for autonomous flight; and iii) introducing a submap-based strategy to mitigate scale drift, which naturally arises in monocular SLAM systems due to calibration inaccuracies and feature-tracking errors. Using the proposed localization framework, we implemented a controller that relies exclusively

on monocular SLAM state estimates to track a pre-planned trajectory.

Experimental results demonstrate that the proposed system enables the drone to navigate the race track autonomously and complete it successfully.

To present our approach, the following sections will describe our methodology and experimental validation, which was conducted on-site in the test arena of season 2 of the A2RL competition.

2 Related Work

Autonomous drone racing (ADR) has emerged as a representative benchmark for agile autonomous flight, attracting significant attention from the robotics community. Early efforts were fostered through the IROS 2016 and 2017 autonomous drone racing competitions, which highlighted the challenges of high-speed navigation in cluttered environments and stimulated research in perception, planning, state estimation, and control [17, 18]. Subsequent large-scale competitions, including the NeurIPS Game of Drones [15] and the AlphaPilot Challenge [10], further accelerated progress by providing standardized simulation environments and real-world racing scenarios for evaluating autonomous flight systems [25].

These initiatives established ADR as a common benchmark for testing autonomous systems under aggressive flight conditions.

Beyond competition settings, ADR has continued to attract broad research interest as a platform for studying vision-based autonomy and high-speed navigation. Many successful systems rely on classical robotics pipelines that combine perception, state estimation, trajectory planning, and control. Recent examples from the A2RL x DCL competition employ monocular visual-inertial state estimation corrected through gate-based localization, together with perception-aware planning and classical control strategies [1]. Similarly, Bosello et al. demonstrated professional-pilot-level performance using a vision-based state-estimation framework that combines gate and keypoint detection and Perspective-n-Point (PnP) localization with Model Predictive Control (MPC) with a time-optimal

trajectory generator, highlighting the effectiveness in an uninstrumented environment [4].

In parallel, ADR research has increasingly explored learning-based solutions. Early work, such as DeepPilot, demonstrated the feasibility of end-to-end neural policies that directly map visual observations to flight commands for autonomous navigation through racing tracks [24]. Subsequent studies investigated learning-based waypoint generation and deep reinforcement learning (RL) policies capable of navigating racing gates at high speed [26, 28].

A major milestone was achieved by Kaufmann et al., who demonstrated a champion-level autonomous racing system based on deep reinforcement learning that successfully competed against expert human pilots in real-world races [13]. This line of research was further extended in later studies comparing optimal-control and RL-based racing strategies, revealing the complementary strengths of model-based and learning-based approaches for aggressive flight [27]. More recently, RL-based systems have incorporated model-based reinforcement learning, domain randomization, and robust monocular perception to improve generalization across environments, simulators, and hardware platforms [29, 8, 3]. A broader overview of learning-based flight control and autonomous racing systems can be found in the coming review paper *Neural Pilots for Agile Flight*, to appear in the 2025 MEXCIR book. Despite these developments, reliable state estimation remains a critical component for enabling robust autonomous flight under aggressive racing conditions.

Visual-Inertial Odometry (VIO) combines camera and IMU measurements to estimate vehicle motion and has become the dominant state-estimation approach for UAV navigation.

Representative systems such as VINS-Mono and OpenVINS achieve accurate real-time localization in GPS-denied environments through tightly coupled visual and inertial fusion [23, 9].

The integration of inertial measurements offers several advantages over purely visual approaches. IMU data provide motion information at significantly higher frequencies than camera sensors, enabling smoother state propagation

between image updates and improving robustness during short-term visual degradation.

Consequently, VIO systems often achieve update rates exceeding those of traditional visual SLAM methods, making them attractive for aggressive flight applications where rapid control updates are required.

Despite these advantages, visual-inertial approaches present several challenges when deployed in high-speed racing scenarios. Accurate performance depends on reliable sensor calibration, including camera-IMU extrinsic parameters, temporal synchronization, and inertial bias estimation. Furthermore, aggressive manoeuvres can generate large accelerations and angular velocities that degrade inertial measurements and increase estimation uncertainty. Recent autonomous racing systems have reported that, despite employing VIO as their primary state-estimation mechanism, additional corrections derived from gate observations are often necessary to compensate for accumulated drift and maintain reliable localization during long flights [13, 1]. These observations suggest that, although VIO currently represents one of the most successful state-estimation solutions for ADR, it remains dependent on inertial sensing quality and, in many cases, on task-specific landmarks for drift correction. This motivates the investigation of alternative approaches capable of exploiting the visual structure of the environment directly.

In particular, visual SLAM offers the possibility of estimating the vehicle state while simultaneously constructing a representation of the surrounding scene, potentially reducing dependence on both inertial measurements and gate-relative localization.

Visual Simultaneous localization and Mapping (SLAM) extends visual odometry by simultaneously estimating vehicle pose while maintaining a map of the environment. Systems such as ORB-SLAM2 and ORB-SLAM3 provide real-time localization, loop closure, relocalization, and map reuse capabilities that significantly reduce long-term drift [19, 5]. Although SLAM has received considerably less attention in ADR than VIO, its ability to exploit general scene structure rather than relying on gate observations

makes it an attractive candidate for more general autonomous racing systems. Furthermore, racing tracks are typically known in advance, enabling maps to be generated offline and reused during flight.

Nevertheless, few studies have investigated whether a monocular SLAM-based pipeline can provide sufficiently accurate and robust state estimation for aggressive autonomous racing without continuous gate-based corrections or tightly coupled visual-inertial fusion.

Recent advances in learning-based perception have improved the robustness of monocular localization systems [6, 33, 32]. Deep-learning methods have been used to estimate depth from RGB images, helping overcome the scale ambiguity of monocular cameras and enhancing SLAM performance [20, 14]. In parallel, visual foundation models have demonstrated powerful feature representations for visual odometry and SLAM, improving feature matching, scene understanding, and motion estimation under challenging conditions [21, 2, 30, 31].

These advances suggest that learned visual representations can complement classical geometric methods, increasing robustness while preserving the consistency and interpretability of model-based localization frameworks.

Despite the significant progress achieved in autonomous drone racing, most state-estimation solutions continue to rely on tightly coupled visual-inertial odometry, gate-based localization, or computationally intensive learning-based approaches.

In contrast, the potential of monocular SLAM as the primary localization mechanism for aggressive autonomous racing remains largely unexplored. This observation motivates the present work, which investigates whether a lightweight monocular SLAM pipeline can provide sufficiently accurate and robust state estimation for autonomous drone racing.

To this end, we develop a complete onboard racing system that combines monocular SLAM, gate-assisted metric scale initialization, submap-based drift mitigation, and a cascade flight-control architecture. The proposed methodology is described in the following section.



Fig. 2. Projection of the known positions of the visible gates onto the camera image plane, depicted in translucent green. The resulting projection is used to generate a synthetic depth image that facilitates feature initialisation in the SLAM system

3 Methodology

The primary concern when developing our aerial system was ensuring we could run all the software components on board the drone. The system considered the following modules: i) visual-based map building, storage, and reload; ii) filter-based state estimation using as input visual SLAM estimates; iii) low-cost control for trajectory tracking using a pre-planned trajectory.

In ADR competitions, so far, the flying arena remains static, meaning that the position and size of the gates will not vary over time. It is also expected to know the size, position, and orientation of the gates, and the order in which these have to be traversed. The proposed system was developed for operation in a flight arena measuring $27 \times 38 \text{ m}^2$. The race track consisted of 10 gate structures, two of which were vertically stacked double gates, yielding a total of 12 gates that the drone was required to traverse.

3.1 Visual SLAM System

We chose to use ORB-SLAM2 in its RGB-D mode [19]. ORB-SLAM2 is a visual SLAM system that relies on the use of binary descriptors to track

image correspondences between frames, from which 3D estimates of map points are calculated. Given the compact nature of the descriptors, the feature tracking process allows the tracking of thousands of features in very few milliseconds.

On the other hand, the RGB-D mode enables feature initialisation with depth information; however, given that the drone is equipped with a monocular camera, features could be created up to scale only. To overcome this issue, we exploited the fact that the arena is known in advance. Thus, at the outset, feature initialisation is carried out by creating a synthetic depth image by back-projecting the 3D shape of each visible gate in the image on its camera plane. Figure 2 shows an example of gates projected on the image plane when the drone is still at the initial point.

Another essential advantage of ORB-SLAM2 is that once initialised, features can be tracked and new features can be created with an empty depth image; that is, each pixel has zero depth. In a previous work, we have shown how this property can be exploited to create and initialise visual features from objects whose known size can be used to form depth images occasionally [16], thus reducing drift when mapping new scene areas.

3.2 SubMapping Strategy

Before the competition, teams are allowed to collect data within the arena. The latter includes recording data from the onboard camera and Inertial Measurement Unit (IMU). Data could be recorded by manual flight or simply by walking around the arena. Therefore, we recorded rosbags from the onboard camera only, which enabled us to build the map by initialising the scale as explained in the previous section. However, due to the size of the arena, we realised that a single map was not feasible given that the map began to drift in scale and shape (bending horizontally) after the fifth gate.

To address the drift, we decided to build submaps of the arena. The first one was initialised with the drone placed at the take-off position. The second map was initialised with the drone placed in front of the sixth gate. Figure 3 shows four columns.

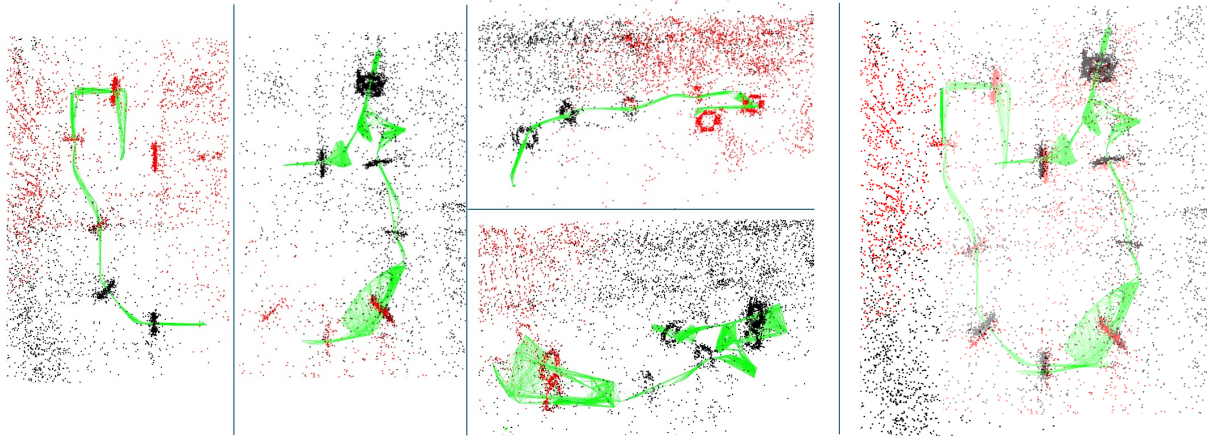


Fig. 3. Top and side views of the two submaps used for localization. The first and second columns show top views of Map 1 (gates 1–6) and Map 2 (gates 6–12), respectively. The third column presents side views of the two maps. The fourth column shows an overlay of both submaps, demonstrating that scale drift remains negligible across the mapped sections of the track

The first column corresponds to a top view of the first map; the first gate is placed at the bottom right of the image, and this map covers up to the sixth gate.

The second column shows a top view of the second map, starting from the sixth gate. The third column shows side views of the two maps: the top image shows the first map and the bottom one shows the second map. The fourth column shows both maps overlapped in top view to show that the scales coincide.

3.3 State Estimation

ORB-SLAM2 provides camera pose estimates represented by a 3D vector for translation and a quaternion for the 3D orientation. In order to estimate other states, such as linear and angular velocity, we implemented an Unscented Kalman Filter (UKF) that used the ORB-SLAM2 estimates as measurements. We chose the UKF to carry out the non-linear estimation given that our process model involved quaternion operations, but also to avoid Jacobian computation [11]. We should also highlight that we decided not to include measurements from the Inertial Measurement Unit (IMU) to avoid the calibration process between the camera and the IMU placement, but also to avoid

dealing with the measurement synchronisation and quick drift in the estimation induced by noise, drone vibration, or incorrect sensor fusion.

To this purpose, we defined the state vector as follows:

$$\mathbf{x}_k = [\mathbf{r}_{WC,k}^\top \quad \mathbf{e}_{WC,k}^\top \quad \mathbf{v}_{WC,k}^\top \quad \boldsymbol{\omega}_{WC,k}^\top]^\top, \quad (1)$$

where $\mathbf{r}_{WC} \in R^3$ denotes the position of the camera with respect to the world frame, $\mathbf{e}_{WC} \in R^3$ is the orientation represented using the exponential map, $\mathbf{v}_{WC} \in R^3$ is the linear velocity, and $\boldsymbol{\omega}_{WC} \in R^3$ is the angular velocity. The resulting state vector has dimension 12.

We use the constant velocity model as the process model defined as:

$$\mathbf{r}_{k+1} = \mathbf{r}_k + (\mathbf{v}_k + \mathbf{w}_{v,k})\Delta t, \quad (2)$$

$$\mathbf{q}_{k+1} = \mathbf{q}_k \otimes \text{Exp}((\boldsymbol{\omega}_k + \mathbf{w}_{\omega,k})\Delta t), \quad (3)$$

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \mathbf{w}_{v,k}, \quad (4)$$

$$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \mathbf{w}_{\omega,k}, \quad (5)$$

where the operator $\text{Exp}(\cdot)$ stands for the Exponential Mapping representation. The process noise terms are modeled as:

$$\mathbf{w}_{v,k} \sim \mathcal{N}(0, \mathbf{Q}_v), \quad (6)$$

$$\mathbf{w}_{\omega,k} \sim \mathcal{N}(0, \mathbf{Q}_\omega). \quad (7)$$

The measurement vector consists of the position and orientation estimates provided by ORB-SLAM. Since the filter state represents orientation using the exponential map, the operator $\text{Exp}(\cdot)$ is used to convert the state orientation $\mathbf{e}_{WC}$ into its corresponding quaternion representation $\mathbf{q}_{WC}$ for comparison with the measured quaternion:

$$\mathbf{z}_k = \begin{bmatrix} \mathbf{r}_{WC,k}^{\text{ORB}} \\ \mathbf{q}_{WC,k}^{\text{ORB}} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{WC,k} \\ \text{Exp}(\mathbf{e}_{WC,k}) \end{bmatrix} + \mathbf{v}_k, \quad (8)$$

$$\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}). \quad (9)$$

3.4 Flight Trajectory

To compute the flight trajectory, we employed the method proposed in [22], specialised to calculate the optimal trajectory to traverse a set of gates by a quadcopter. The method produces a set of 3D coordinates defining the flight path. As input, this method expects the center of the gates as a 3D point, although it admits additional 3D waypoints to customize the shape of the trajectory. It also considers the drone's weight, arm length, propeller size, and type (two or three blades). Based on that, a path is calculated with translation, orientation, linear, and body velocities:

$$\mathbf{T}_{\text{path}} = [\mathbf{p}^T \quad \mathbf{v}^T \quad \mathbf{v}_b^T \quad \boldsymbol{\omega}^T \quad \mathbf{q}^T]^T, \quad (10)$$

where:

$$\mathbf{p} = [x \quad y \quad z]^T, \quad (11)$$

$$\mathbf{v} = [v_x \quad v_y \quad v_z]^T, \quad (12)$$

$$\mathbf{v}_b = [b_x \quad b_y \quad b_z]^T, \quad (13)$$

$$\boldsymbol{\omega} = [w_x \quad w_y \quad w_z]^T, \quad (14)$$

$$\mathbf{q} = [q_x \quad q_y \quad q_z \quad q_w]^T. \quad (15)$$

We emphasize that a set of waypoints was used to generate a suitable flight trajectory while avoiding aggressive maneuvers.

This design choice was based on the maximum speed that our controller was known to achieve reliably. Additionally, a special maneuver was planned between Gates 5 and 6: the drone first performs a lateral motion until it is properly

aligned with Gate 6, after which it resumes the nominal trajectory. Regarding the body velocities $\mathbf{v}_b$, it is worth noticing that these velocities are calculated considering the drone's dynamics, such that if the drone has to perform a turn, then this velocity is planned to decrease at a certain position in the trajectory to enable the drone to reduce speed and achieve the turn. Thus, this trajectory planner, besides providing a feasible flight trajectory, also provides valuable speed information to be considered by the controller.

Figure 4 illustrates the planned trajectory. The left image presents a top view of the trajectory, while the right image shows a perspective view of the arena, where the gates and the planned trajectory can be observed together. Note that there are two double gates, the first one where a Split-S manoeuvre is demanded and a second double gate where a ladder-like trajectory must be performed.

4 Control

The drone is controlled by sending angle rate commands to the flight controller. To this end, we designed a set of controllers to enable altitude control given a reference height and trajectory tracking.

4.1 Altitude Controller

The altitude control strategy is implemented as a cascaded controller composed of an outer position loop and an inner velocity loop. The objective of the outer loop is to generate a desired vertical velocity based on the altitude error, while the inner loop tracks this velocity reference by adjusting the collective thrust. This separation of the position and velocity dynamics simplifies controller tuning, improves stability and transient response, and naturally limits ascent and descent rates, resulting in smoother and more robust altitude regulation than a single PID acting directly on the altitude error.

Therefore, the altitude error is defined as:

$$e_z = z_r - z, \quad (16)$$

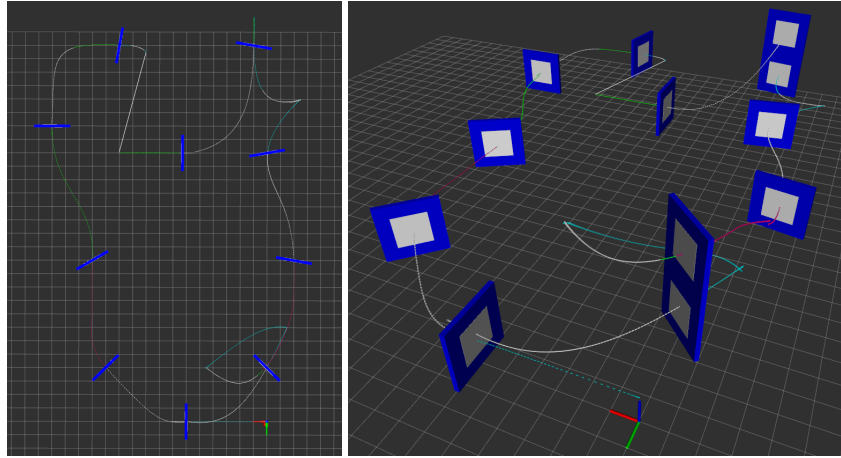


Fig. 4. Top and side views of the planned trajectory. The course includes two double-gate sections that require complex three-dimensional manoeuvres. The first double gate is traversed using a Split-S manoeuvre, while the second requires the drone to cross the upper gate, descend, and subsequently reposition itself to face and traverse the lower gate, producing a ladder-like trajectory

where z_r is the reference altitude (defined by the user or retrieved from the planned trajectory) and z is the current drone's altitude.

A proportional controller is used in the outer loop to generate the desired vertical velocity:

$$v_{z,r} = K_z e_z = K_z (z_r - z), \quad (17)$$

where K_z is the proportional gain of the altitude controller.

The inner-loop controller is represented by the PID operator:

$$\text{PID}(r, y) = K_p(r - y) + K_i \int (r - y) dt + K_d \frac{d}{dt}(r - y), \quad (18)$$

where r and y denote the reference and measured signals, respectively, and K_p , K_i , and K_d are the proportional, integral, and derivative gains.

The thrust correction signal is then computed as:

$$u_z = \text{PID}(v_{z,r}, v_z), \quad (19)$$

where v_z is the measured vertical velocity.

Finally, the total commanded thrust is obtained by adding the control correction to the nominal hover thrust T_0 :

$$T = T_0 + u_z. \quad (20)$$

This cascaded architecture allows the outer loop to regulate altitude while the faster inner loop tracks the desired vertical velocity. As thrust acts directly on the vehicle's vertical dynamics through acceleration, introducing a velocity control loop improves tracking performance and disturbance rejection while simplifying the regulation of altitude.

4.2 Trajectory Tracking Controller

Given the current position of the vehicle, the closest point on the reference trajectory is determined together with a look-ahead offset. The trajectory query returns the tuple:

$$[\mathbf{p}_i, v_r, \mathbf{d}_i], \quad (21)$$

where $\mathbf{p}_i$ is the selected point on the trajectory, v_r is the desired forward velocity expressed in the body frame, and $\mathbf{d}_i$ is the trajectory direction vector at that point.

The pitch controller is implemented as a cascaded velocity–attitude controller. First, a desired longitudinal acceleration is computed from the velocity tracking error using the PID operator:

$$a_{\theta,d} = \text{PID}(v_r, v), \quad (22)$$

where v denotes the measured forward body velocity.

Assuming small roll angles, the desired acceleration is converted into a desired pitch angle according to:

$$\theta_d = \text{clamp} \left(\tan^{-1} \left(\frac{a_{\theta,d}}{g} \right), -\theta_{\max}, \theta_{\max} \right), \quad (23)$$

where g is the gravitational acceleration and $\theta_{\max}$ is the maximum allowable pitch angle.

The pitch tracking error is then defined as:

$$e_{\theta} = \theta_d - \theta, \quad (24)$$

where θ is the measured pitch angle. Finally, a proportional controller generates the pitch control command:

$$u_{\theta} = K_{\theta} e_{\theta}. \quad (25)$$

The roll controller is responsible for minimizing the lateral deviation of the vehicle with respect to the reference trajectory. Let $\mathbf{r}$ denote the current position of the vehicle and let $\mathbf{q}(\mathbf{e})$ be the quaternion representation of its orientation, obtained from the exponential-map representation $\mathbf{e}$ (these are estimates obtained from the UKF).

Given the selected trajectory point $\mathbf{p}_i$, the relative position between the trajectory and the vehicle is first expressed in the body frame as:

$$\mathbf{p}_T^c = \mathbf{R}(\mathbf{q})^T (\mathbf{p}_i - \mathbf{r}), \quad (26)$$

where $\mathbf{R}(\mathbf{q})$ is the rotation matrix associated with the quaternion $\mathbf{q}$.

The lateral distance to the trajectory is then defined as the body-frame y -coordinate of the transformed point:

$$d_T = -\mathbf{p}_{T,y}^c, \quad (27)$$

where the sign convention is chosen such that positive values indicate that the vehicle must generate a corrective lateral acceleration toward the trajectory.

A desired lateral acceleration is computed through the PID controller:

$$a_{\phi,d} = \text{PID}(0, d_T), \quad (28)$$

where the reference value is zero since the objective is to eliminate the lateral tracking error.

Assuming small pitch angles, the desired lateral acceleration is converted into a desired roll angle according to:

$$\phi_d = \text{clamp} \left(\tan^{-1} \left(\frac{a_{\phi,d}}{g} \right), -\phi_{\max}, \phi_{\max} \right), \quad (29)$$

where g is the gravitational acceleration and $\phi_{\max}$ is the maximum allowable roll angle.

The roll tracking error is defined as:

$$e_{\phi} = \phi_d - \phi, \quad (30)$$

where ϕ is the measured roll angle. Finally, the roll control command is generated by a proportional controller:

$$u_{\phi} = K_{\phi} e_{\phi}. \quad (31)$$

The yaw controller is responsible for aligning the vehicle heading with the orientation specified by the reference trajectory. For each selected trajectory point, the trajectory query returns a direction vector $\mathbf{d}_i$ indicating the desired heading of the vehicle at that location.

The current heading direction of the vehicle is obtained from its orientation quaternion $\mathbf{q}$. Let:

$$\mathbf{e}_x = [1 \ 0 \ 0]^T, \quad (32)$$

denote the unit vector aligned with the body-frame forward axis. The corresponding direction expressed in the world frame is:

$$\mathbf{d}_c = \mathbf{R}(\mathbf{q})\mathbf{e}_x, \quad (33)$$

The heading error is computed as the signed angle between the current direction vector $\mathbf{d}_c$ and the desired trajectory direction $\mathbf{d}_i$. Restricting the computation to the horizontal plane, the angular error is given by:

$$e_{\psi} = \text{atan2} (d_{c,x}d_{i,y} - d_{c,y}d_{i,x}, d_{c,x}d_{i,x} + d_{c,y}d_{i,y}). \quad (34)$$

This expression computes the signed angle between the two vectors using their cross and dot products, yielding an error in the interval $[-\pi, \pi]$.

The yaw control command is then obtained directly from the PID controller:

$$u_{\psi} = \text{PID}(0, e_{\psi}), \quad (35)$$

where the reference value is zero since the objective is to align the vehicle heading with the desired trajectory direction.

5 Experiments

The aerial system developed in this work was evaluated throughout the selection stages of the A2RL Drone Competition. Its performance enabled our team to advance to the Third Selection Phase, held on-site in Abu Dhabi, and subsequently to participate in the Fourth Selection Phase, which served as the qualification stage for the final competition. The results presented in this section correspond to the system's performance during this final selection stage.

The aerial system developed in this work was deployed on a quadcopter built on a custom carbon-fiber frame provided by the competition organizers. The platform was equipped with a flight controller, a monocular rolling-shutter camera with a 155° field of view, an IMU, and an NVIDIA Orin NX companion computer with GPU acceleration.

The total takeoff weight of the vehicle was 970 g. The onboard software was implemented using the Robot Operating System (ROS) Noetic running on Ubuntu 20.04 LTS. All perception, planning, and control modules were executed onboard the Orin NX, without reliance on external communication. The control commands computed by the controller were sent directly to the flight controller through a wired connection.

Our system successfully completed the track eight times, including one test in which a double lap was attempted. During this test, the drone collided with the first gate after successfully crossing 24 gates. Nevertheless, these experiments demonstrated reliable navigation performance, with seamless map switching throughout the flight. Prior to each flight test, the maps were loaded from disk into the system's memory. Upon crossing the fifth gate, the system switched to the second map and triggered the relocalization mechanism,

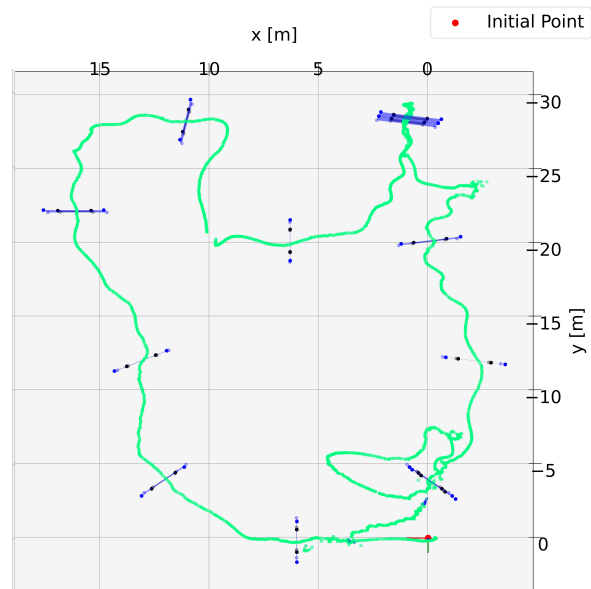


Fig. 5. Top view of a representative successful trajectory, selected from ten autonomous flights in which the drone successfully completed a full lap of the course

which rapidly estimated the drone's pose within the new map. Although a slight discrepancy may have existed between this estimate and the drone's last estimated position at the time of relocalization, UKF effectively mitigated any resulting discontinuity, ensuring smooth state estimation and stable flight.

Figure 5 presents a top view of one of the recorded trajectories, shown in green, to illustrate the performance of the proposed system. As can be observed, the drone successfully traversed all gates, including the two double-gate sections. Figure 6 shows a three-dimensional side view of the same trajectory, with the path color-coded according to the drone's speed throughout the flight. The starting point is represented by a 3D coordinate frame located at $(0, 0, 0)$. It can be observed that the drone attained its highest speed while approaching and traversing the double-gate sections.

To better illustrate the altitude variations during autonomous operation, Figure 7 presents the drone's height profile throughout the flight. Several peaks can be observed, primarily due to

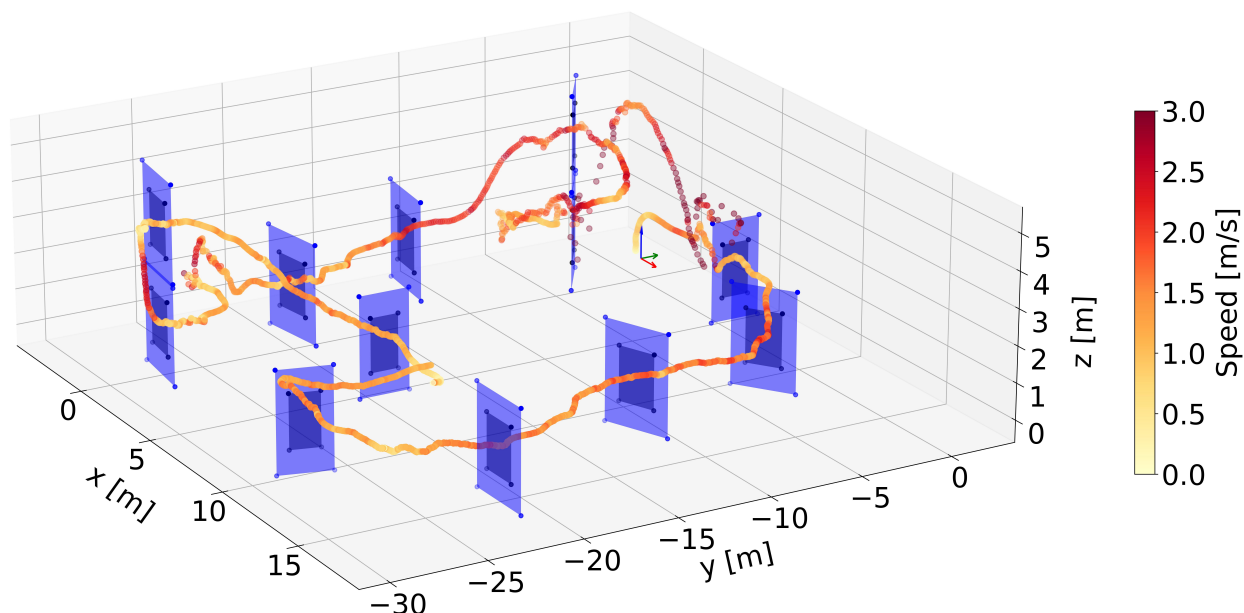


Fig. 6. Three-dimensional view of the trajectory shown in Figure 5, colour-coded according to the drone's speed along the track. The highest speeds are attained while approaching and traversing the double-gate sections

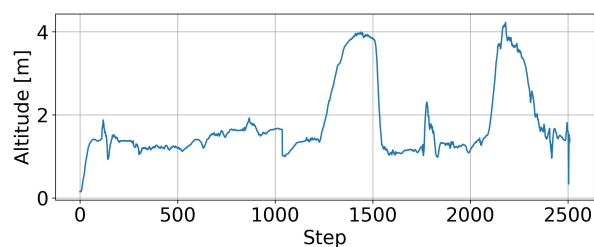


Fig. 7. Altitude profile of the drone along the flight trajectory shown in Figure 5

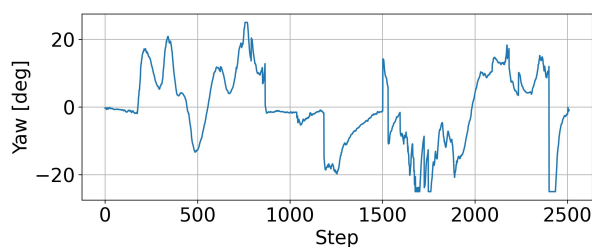


Fig. 8. Yaw angle (heading) profile of the drone along the flight trajectory shown in Figure 5

disturbances encountered during navigation. For example, minor altitude deviations occurred when the drone passed very close to certain gates, likely due to aerodynamic effects. Another noticeable fluctuation appears between steps 1500 and 2000, when the drone traversed the first double gate.

This maneuver was executed by flying backward through the lower section of the double gate rather than performing a turn toward it, since direct gate observation was not required for state estimation.

After crossing the gate, the drone rotated toward the ninth gate to continue the trajectory. This region

of the arena presented particularly challenging visual conditions, characterized by dim illumination and a large black curtain covering the southern side of the venue.

These conditions degraded feature-tracking performance and introduced temporary instabilities in the SLAM-based state estimates. Nevertheless, the filtering stage effectively attenuated these disturbances, resulting in stable state estimation and successful completion of the maneuver.

Figure 8 shows the evolution of the drone's yaw angle during the flight.

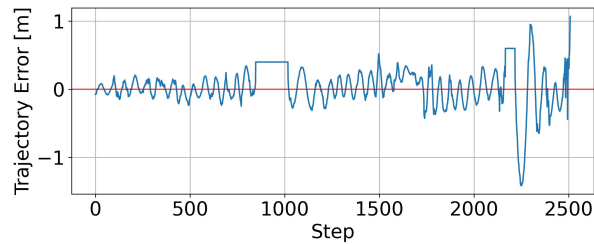


Fig. 9. Trajectory tracking error relative to the planned trajectory for the flight shown in Figure 5

Consistent with the altitude profile, small heading perturbations are visible, particularly between the first double gate and the ninth gate. This section of the course was especially demanding, requiring the drone to execute a zig-zag maneuver, ascend to traverse the upper section of the second double gate, and subsequently descend and maneuver around the structure to cross its lower section.

The yaw profile reflects the orientation changes associated with these maneuvers and shows that the drone ultimately returned to its initial heading of approximately zero degrees upon completing the lap.

Figure 9 presents the trajectory tracking error throughout the flight. It can be observed that the error remained close to zero for most of the trajectory, reaching a maximum absolute error of 1.4. It is important to note, however, that this peak error occurred after the drone had crossed the upper section of the second double gate and was descending towards its lower section before passing through it to complete the lap.

This manoeuvre was particularly challenging. The drone traversed the upper section of the double gate while flying forwards, but during the subsequent descent, it maintained its forward-facing orientation while flying backwards towards the lower section of the gate.

This strategy avoided the need for a complex turning manoeuvre, which could have increased the risk of visual tracking degradation and potential SLAM failure. Despite the increased difficulty of this segment, the tracking error remained bounded, demonstrating the robustness of the proposed navigation and state-estimation framework.

Finally, the complete state-estimation and control loop operated at an average frequency of 20 Hz throughout the autonomous flights. No GPU acceleration was employed during operation, which contributed to reduced power consumption and increased flight endurance. This consideration was particularly important when using batteries with degraded performance or batteries that were not fully charged, as GPU utilisation significantly increased the rate of battery discharge.

A video demonstrating the successful execution of two consecutive laps is available (see footnote¹).

6 Conclusion

We have presented an aerial system for Autonomous Drone Racing (ADR), whose deployment was successfully validated during the final qualification phase of Season 2 of the A2RL Drone Competition, held in Abu Dhabi. The proposed approach relied exclusively on visual sensing, with all perception, state-estimation, and control processes executed onboard the drone, without requiring GPU acceleration during autonomous flight.

The system was built upon a monocular SLAM framework employing a submapping strategy that leveraged the ability to generate maps offline, load them prior to flight and switch between them seamlessly. This approach effectively mitigated scale drift and enabled accurate pose estimation throughout the race track. The resulting pose estimates were incorporated as measurements into an Unscented Kalman Filter (UKF), which provided robust estimates of the drone's position, orientation, and velocity. These state estimates were subsequently used by the flight controller to achieve reliable autonomous navigation. Experimental results demonstrated the effectiveness of the proposed system, which successfully completed the race track in eight test flights, including one experiment in which two consecutive laps were completed.

Rather than targeting the highest possible racing speeds, this work investigated the feasibility of monocular SLAM as a lightweight and more

¹<https://youtu.be/u6YxCF7HMsE>

generalisable alternative for ADR under reduced environmental assumptions. In particular, the proposed approach avoids reliance on IMU calibration and gate-relative localization, whilst retaining the ability to navigate using general scene structure. We argue that the presented results demonstrate the potential of computationally efficient SLAM systems for ADR, particularly in resource-constrained platforms employing low-cost onboard computers and flight controllers. Furthermore, unlike approaches that require carefully controlled illumination conditions and highly salient gate appearances to guarantee robust perception, the proposed method offers greater potential for deployment in broader and less constrained environments.

Future work will focus on investigating emerging SLAM methodologies that offer improved computational efficiency while remaining suitable for deployment on embedded platforms. We are confident that advances in visual SLAM will play a key role in enabling ADR systems to operate reliably in increasingly challenging environments, including low-light and outdoor scenarios, enabling autonomous navigation in a manner more closely resembling that of human pilots, who often rely on sparse visual cues and broader scene understanding to maintain high-speed flight.

References

1. **Azhari, M. B., Han, D., You, J. I., Park, S., Shim, D. H. (2025).** Drift-corrected monocular vio and perception-aware planning for autonomous drone racing.
2. **Azhari, M. B., Shim, D. H. (2025).** Dino-vo: A feature-based visual odometry leveraging a visual foundation model. *IEEE Robotics and Automation Letters*, Vol. 10, No. 9, pp. 9152–9159. DOI: 10.1109/LRA.2025.3592065.
3. **Bahnam, S. A., Ferede, R., Blaha, T. M., Lang, A. E., Lucassen, E., Missinne, Q., Verraest, A. E. C., Wagter, C. D., de Croon, G. C. H. E. (2026).** Monorace: Winning champion-level drone racing with robust monocular ai.
4. **Bosello, M., Pinzarrone, F., Kiade, S., Aguiari, D., Keuter, Y., AlShehhi, A., Caminati, G., Wong, K. L., Chou, K. S., Halepota, J., Alneyadi, F., Panerati, J., Pau, G. (2026).** On your own: Pro-level autonomous drone racing in uninstrumented arenas. *IEEE Robotics and Automation Letters*, Vol. 11, No. 3, pp. 2674–2681. DOI: 10.1109/lra.2026.3653405.
5. **Campos, C., Elvira, R., Gomez, J. J., Montiel, J. M. M., Tardós, J. D. (2021).** ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *IEEE Transactions on Robotics*, Vol. 37, No. 6, pp. 1874–1890.
6. **Clark, R., Wang, S., Wen, H., Markham, A., Trigoni, N. (2017).** Vinet: visual-inertial odometry as a sequence-to-sequence learning problem. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI Press, pp. 3995–4001.
7. **De Wagter, C., Paredes-Vallés, F., Sheth, N., de Croon, G. C. (2022).** The sensing, state-estimation, and control behind the winning entry to the 2019 artificial intelligence robotic racing competition. *Field Robotics*, Vol. 2, pp. 1263–1290.
8. **Ferede, R., Blaha, T., Lucassen, E., De Wagter, C., De Croon, G. C. (2025).** One net to rule them all: Domain randomization in quadcopter racing across different platforms. *2025 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 6357–6363.
9. **Geneva, P., Eckenhoff, K., Lee, W., Yang, Y., Huang, G. (2020).** OpenVINS: A research platform for visual-inertial estimation. *Proc. of the IEEE International Conference on Robotics and Automation*, Paris, France.
10. **Guerra, W., Tal, E., Murali, V., Ryou, G., Karaman, S. (2019).** Flightgoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality. *2019 IEEE/RSJ International*

- Conference on Intelligent Robots and Systems (IROS), pp. 6941–6948. DOI: 10.1109/IROS40897.2019.8968116.
11. Julier, S. J., Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, Vol. 92, No. 3, pp. 401–422.
 12. Jung, S., Cho, S., Lee, D., Lee, H., Shim, D. H. (2018). A direct visual servoing-based framework for the 2016 iros autonomous drone racing challenge. *Journal of Field Robotics*, Vol. 35, No. 1, pp. 146–166.
 13. Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., Scaramuzza, D. (2023). Champion-level drone racing using deep reinforcement learning. *Nature*, Vol. 620, No. 7976, pp. 982–987. DOI: 10.1038/s41586-023-06419-4.
 14. Liu, F., Huang, M., Ge, H., Tao, D., Gao, R. (2024). Unsupervised monocular depth estimation for monocular visual slam systems. *IEEE Transactions on Instrumentation and Measurement*, Vol. 73, pp. 1–13. DOI: 10.1109/TIM.2023.3342210.
 15. Madaan, R., Gyde, N., Vemprala, S., Brown, M., Nagami, K., Taubner, T., Cristofalo, E., Scaramuzza, D., Schwager, M., Kapoor, A. (2020). Airsim drone racing lab. *arXiv preprint arXiv:2003.05654*.
 16. Martinez-Carranza, J., Hernández-Farías, D. I., Rojas-Perez, L. O., Cabrera-Ponce, A. A. (2023). Language meets yolov8 for metric monocular slam. *Journal of real-time image processing*, Vol. 20, No. 4, pp. 59.
 17. Moon, H., Martinez-Carranza, J., Cieslewski, T., Faessler, M., Falanga, D., Simovic, A., Scaramuzza, D., Li, S., Ozo, M., De Wagter, C., de Croon, G., Hwang, S., Jung, S., Shim, H., Kim, H., Park, M., Au, T.-C., Kim, S. J. (2019). Challenges and implemented technologies used in autonomous drone racing. *Intelligent Service Robotics*, Vol. 12, No. 2, pp. 137–148. DOI: 10.1007/s11370-018-00271-6.
 18. Moon, H., Sun, Y., Baltes, J., Kim, S. J. (2017). The iros 2016 competitions [competitions]. *IEEE Robotics Automation Magazine*, Vol. 24, No. 1, pp. 20–29. DOI: 10.1109/MRA.2016.2646090.
 19. Mur-Artal, R., Tardós, J. D. (2017). ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, Vol. 33, No. 5, pp. 1255–1262. DOI: 10.1109/TR0.2017.2705103.
 20. Nawal, M., Brahmanage, G., Leung, H. (2024). Rgb-pd slam: Scale consistent monocular slam using predicted depth. *2024 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan)*, pp. 359–360. DOI: 10.1109/ICCE-Taiwan62264.2024.10674415.
 21. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.-Y., Li, S.-W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., Bojanowski, P. (2024). Dinv2: Learning robust visual features without supervision.
 22. Qin, C., Michet, M. S., Chen, J., Liu, H. H.-T. (2024). Time-optimal gate-traversing planner for autonomous drone racing. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 8693–8699.
 23. Qin, T., Li, P., Shen, S. (2018). Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, Vol. 34, No. 4, pp. 1004–1020. DOI: 10.1109/TR0.2018.2853729.
 24. Rojas-Perez, L. O., Martinez-Carranza, J. (2020). DeepPilot: A cnn for autonomous drone racing. *Sensors*, Vol. 20, No. 16. DOI: 10.3390/s20164524.
 25. Rojas-Perez, L. O., Martinez-Carranza, J. (2021). On-board processing for autonomous drone racing: An overview. *Integration*, Vol. 80,

- pp. 46–59. DOI: <https://doi.org/10.1016/j.vlsi.2021.04.007>.
26. **Rojas-Perez, L. O., Martinez-Carranza, J. (2022).** Where are the gates: Discovering effective waypoints for autonomous drone racing. *Advances in Artificial Intelligence – IBERAMIA 2022: 17th Ibero-American Conference on AI*, Cartagena de Indias, Colombia, November 23–25, 2022, Proceedings, Springer-Verlag, Berlin, Heidelberg, pp. 353–365. DOI: 10.1007/978-3-031-22419-5_30.
 27. **Song, Y., Romero, A., Müller, M., Koltun, V., Scaramuzza, D. (2023).** Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. *Science Robotics*, Vol. 8, No. 82. DOI: 10.1126/scirobotics.adg1462.
 28. **Song, Y., Steinweg, M., Kaufmann, E., Scaramuzza, D. (2021).** Autonomous drone racing with deep reinforcement learning. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE Press, pp. 1205–1212. DOI: 10.1109/IROS51168.2021.9636053.
 29. **Verraest, A., Bahnam, S., Ferede, R., de Croon, G., De Wagter, C. (2025).** Skydreamer: Interpretable end-to-end vision-based drone racing with model-based reinforcement learning. arXiv preprint arXiv:2510.14783.
 30. **Wang, J., Chen, M., Karaev, N., Vedaldi, A., Rupprecht, C., Novotny, D. (2025).** Vggt: Visual geometry grounded transformer. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
 31. **Wang, J., Chen, M., Zhang, S., Karaev, N., Schönberger, J., Labatut, P., Bojanowski, P., Novotny, D., Vedaldi, A., Rupprecht, C. (2026).** Vggt- ω .
 32. **Xolo-Tlapanco, N., Martinez-Carranza, J. (2025).** 3d cnn and multi-head attention to relative camera position. **Martinez-Carranza, J.**, editor, 16th International Micro Air Vehicle Conference and Competition, San Andrés Cholula, Puebla, Mexico, pp. 18–24. Paper no. IMAV2025-1.
 33. **Yen-Chen, L., Florence, P., Barron, J. T., Rodriguez, A., Isola, P., Lin, T.-Y. (2021).** inerf: Inverting neural radiance fields for pose estimation. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1323–1330. DOI: 10.1109/IROS51168.2021.9636708.

Article received on 01/03/2026; accepted on 18/07/2026.

*Corresponding author is Jose Martinez-Carranza.