Special Issue: The Mohamed Bin Zayed International Robotics Challenge (MBZIRC) 2020

Regular Article

Autonomous Cooperative Multi-Vehicle System for Interception of Aerial and Stationary Targets

Lima Agnel Tony[1], Shuvrangshu Jana[1], Varun V. P.[2], Aashay Anil Bhise[1], Aruul Mozhi Varman S.[2], Vidyadhara B. V.[1], Mohitvishnu S. Gadde[1], Raghu Krishnapuram[2] and Debasish Ghose[1]

[1]Guidance, Control, and Decision Systems Laboratory (GCDSL), Department of Aerospace Engineering, Indian Institute of Science, Bangalore-560012, India.

[2]Robert Bosch Center for Cyber-Physical Systems, Indian Institute of Science, Bangalore-560012, India.

Abstract: This paper presents the design, development, and testing of hardware-software systems by the IISc-TCS team for Challenge 1 of the Mohamed Bin Zayed International Robotics Challenge 2020. The goal of Challenge 1 was to grab a ball suspended from a maneuvering UAV and pop balloons anchored to the ground, using suitable manipulators. The important tasks carried out to address this challenge include the design and development of a hardware system with efficient grabbing and popping mechanisms, considering the restrictions in volume and payload, design of accurate target interception algorithms using visual information suitable for outdoor environments, and development of a software architecture for dynamic, multi-agent, aerial systems performing complex tasks. In this paper, we discuss the design of a custom end-effector mounted on a single degree of freedom manipulator, and robust algorithms for the interception of targets in an uncertain environment. Vision-based guidance and tracking strategies are developed based on the concept of pursuit engagement and artificial potential function. The software architecture presented in this work develops an Operation Management System (OMS) architecture that allocates static and dynamic tasks collaboratively among multiple UAVs to perform any given task. An important aspect of this work is that all the systems developed were designed to operate in completely autonomous mode. A detailed description of the architecture along with simulations of complete challenge in the Gazebo environment and field experiment results are also included in this work. The developed hardware-software system is useful for counter-UAV systems and can be used for other applications,

Keywords: cooperative robots, computer vision, motion planning, obstacle avoidance, planning

1. Introduction

Autonomous systems are becoming an integral part of human society —due to rapid advancements in hardware, software, and algorithms —and their apparent utility in many important applications.

Received: 30 September 2020; revised: 26 May 2021; accepted: 6 July 2021; published: 15 March 2022.

Correspondence: Debasish Ghose, Guidance, Control, and Decision Systems Laboratory (GCDSL), Department of Aerospace Engineering, Indian Institute of Science, Bangalore-560012, India., Email: dghose@iisc.ac.in

Table 1. MBZIRC 2020 Challenge 1 specifications

Parameter	Value	Remarks
Arena	100 m × 40 m × 20 m	Covered with net
Target UAV trajectory	Repetitive figure-of-eight	Altitude: 5 m - 15 m
Figure-of-eight spec.	Any orientation	Center fixed
Target UAV speed	3 m/s or 7 m/s	First 7 min/rest 8 min
Ball color	Yellow	Softball
Ball specifications	10 cm dia., 60 g	-
String for ball	1.45 m	Steel wire
Balloon specifications	60 cm dia., Green color	Altitude: 5 m
Balloon rod	1.5 m from ground	5 nos., randomly placed
Time	15 min	5 min to set up

As a result, autonomous and semi-autonomous robots have been quietly replacing humans in many daily tasks, that were repetitive in nature. Aerial robotics is one such evolving field where many complex applications are being addressed by researchers worldwide. Mohamed Bin Zayed International Robotics Challenge (Khalifa, 2020) provides exciting problems to roboticists. The tasks involved in this competition provide solutions to many practical applications involving aerial and ground robots. MBZIRC 2020 poses three problems: aerial and ground target interception, wall construction, and indoor and outdoor firefighting. This paper presents a complete solution to the problems in Challenge 1, which involves autonomously searching, detecting, and grabbing a maneuvering target as the primary task. The secondary task is to search, detect, and burst randomly-located stationary targets.

The main goal of Challenge 1 is to grab a ball hanging from a moving target drone. The target moves continuously in a figure-of-eight path. The figure-of-eight can be in any orientation in space. The ball is yellow and 10 cm in diameter. The drone carrying the ball travels at a maximum speed of 10 m/s. Three UAVs that satisfy the volume constraint of 1.2 m × 1.2 m × 0.5 m can be used for the task. Before take-off, the UAV should satisfy the volume constraint but can extend appendages in the air. For safety reasons, the cap on the UAV speed is 30 km/h. The UAVs are assigned to take-off from a marked 10 m × 10 m square in the middle of the arena. The UAV should autonomously detect the ball, approach, grab the ball hanging from the target drone and land back in the 10 m × 10 m area. The grabbing force required is less than 4 N. The second Challenge 1 problem is the popping of balloons attached to the top of 1.5 m high poles. The balloons are 65 cm in diameter and green in color. A maximum of five balloons are placed at various random locations within the arena. The challenge is to be completed within 15 minutes. The summary of Challenge 1 specifications are given in Table 1.

For the successful execution of Challenge 1, the primary building blocks are aerial manipulator hardware design, target detection, tracking, aerial interception, and coordination among the agents through mission planner. An aerial manipulator's design for grasping a dynamic aerial target is challenging due to various issues such as stability, control, and accuracy requirements. Moreover, aerial manipulation with UAVs has varied complexities. Instability during manipulation due to forces and moments, non-linear coupled dynamics and payloads constraints are a few among them.

In this paper, the above-mentioned aspects are addressed by developing an efficient and stable aerial manipulation mechanism, robust and computationally efficient tracking and grabbing algorithms using visual information, and a multi-agent software architecture, called Operation Management System (OMS), for allocating tasks to UAVs, ensuring zero conflicts and maximum safety. A custom-designed and manufactured passive end-effector with linearly actuated arm is attached to a hexacopter for grabbing and interception. A 3D guidance strategy based on the idea of pursuit guidance is developed for the interception by using visual information from a monocular camera. An adaptive controller is used to handle the uncertainties in system dynamics during the grabbing phase and wind disturbance. A computationally efficient and robust vision algorithm is also developed, that provides precise detection of the target under varied lighting conditions, thus improving the algorithms' performance. The OMS is developed using a multi-master network in such a way that many critical tasks such as collision avoidance, are addressed in high priority mode, without affecting the primary task, such as grabbing. Exploring, tracking, and grabbing tasks are designed for multi-UAV collaboration without compromising the overall system's efficiency or redundancy. All these algorithms were tested on Gazebo, a rigid-body physics simulator, taking care to simulate the physical and sensor parameters such as a manipulator attachment, the target ball hanging from the target UAV, balloons, and camera vision parameters. The complete Challenge 1 mission is simulated in the Gazebo environment using the software architecture developed on ROS. The functionality of individual algorithms are first checked through simulations, by incorporating models of sensors and manipulation mechanism in the virtual environment. The developed hardware and software solutions are tested outdoors after successful Software-In-The-Loop simulations (SITL) and Hardware-In-The-Loop simulations (HITL). The paper reports flight-test results and highlights significant design features.

The remainder of this report is organized as follows: Section 2 gives a detailed literature survey. Section 3 presents the major tasks involved, and Section 4 presents the algorithms developed to address these. Section 5 describes the mission modes associated with the software architecture, while Section 6 introduces the OMS and describes it in detail. Section 7 presents the ROS simulations. Section 8 details the hardware design, and Section 9 brings out the experimental results. A brief discussion about this challenge and developed OMS framework is given in Section 10 and Section 11 concludes the paper.

2. Relevant Literature

In this section, we briefly survey the literature that addresses the problems related to some components of Challenge 1. A comprehensive survey of aerial manipulation mechanisms and UAV platforms, along with design methods, are given in (Khamseh et al., 2018). UAV manipulators for specific purposes like object extraction, inspection and transportation are described in (Suarez et al., 2016; Ramon Soria et al., 2016; Suarez et al., 2019; Suarez et al., 2020). Design of aerial grasping system is reported in (Papachristos et al., 2014; Ramon-Soria et al., 2020); but only for stationary objects. A multi-link manipulator will have a high variation of center of gravity and moment of inertia during the manipulation mechanism movement than a body-fixed manipulator and can have stability issues during grabbing from a flying target. The manipulator's design as a part of the UAV frame is reported in several works (Tsukagoshi et al., 2015; Alexis et al., 2016). However, these are not feasible solutions for Challenge 1 as the vehicle might crash due to the steel wire getting entangled in the propellers while attempting to grab the target. A manipulator as a rigid tool attached to the UAV is reported in (Nguyen et al., 2015; Gioioso et al., 2014; Fumagalli et al., 2014), which are designed to interact with the physical environment. In (Gioioso et al., 2014), a quadrotor system is designed to exert force on its surrounding environment, at near hovering conditions. In (Papachristos et al., 2014), a similar application is presented, where the thrust vectoring capability of a Tri-Tiltrotor UAV is utilized to achieve manipulation. There was no restriction on the maximum size in all these cases, and the relative motion between the manipulator's end effector and the target object is not aggressive.

Visual object tracking is crucial in semantic video interpretation, and several different approaches have been developed in recent years. Correlation filters were used to model the visual appearance of an arbitrary target, and correlation operation was used as an effective approach to achieving tracking-by-detection (Bolme et al., 2010). Numerous techniques have been developed to improve the computational efficiency and tracking capabilities of a correlation filter-based tracker by employing multi-channel versions (Galoogahi et al., 2013), spatial boundaries (Danelljan et al., 2015), deep features (Danelljan et al., 2017a), and particle filter based methods (Zhang et al., 2017). The correlation filter based approaches are simple to implement and computationally lighter. Though these approaches can run in real time on most modern hardware, they are limited in handling occlusions, scale variations, and variations in appearance. Considering their limitations in object-detection, these approaches are not adequately equipped to track high speed objects. Hence, a two-step approach to tracking high speed targets with deep neural networks to perform object-detection is employed to overcome the above-mentioned problems (Bewley et al., 2016; Wojke et al., 2017). The first step involves detecting and localizing all the objects of interest in the image using object detectors such as TinyYOLOv3 (Redmon and Farhadi, 2018). The second step is to uniquely identify the actual objects corresponding to the detections and associate their newer detections with their trackers over time. A Kalman filter is used to maintain the track, and the Hungarian algorithm is used to accomplish the association task. Object re-identification features (Re-ID features) (Liu et al., 2012) and Intersection over Unions (IoU) or Mahalanobis distance among the detections and trackers' predictions are used to compute the cost matrix to associate detections to object trackers across multiple frames.

Traditionally, handcrafted features like color histogram (Xiong et al., 2014), HOG (Danelljan et al., 2017b), and dense SIFT (Zhao et al., 2013) were used as Re-ID features. With advancements in deep learning, the regions corresponding to the bounding boxes are fed through an identity embedding deep neural network (Li et al., 2014) to learn the Re-ID features. These deep learning based object tracking techniques typically require two deep models addressing object-detection and Re-ID feature computation. Considering the fact that all the UAV detections are used in the initial phases to search for the attached target ball and their information is not critical once the target ball is located, the computational complexity involved in using Re-ID features in the association metric out-weighs its benefits in our case. Also, the Re-ID features are not helpful in balloon detection and tracking as the balloons are similar in size and appearance.

In general, vision-based interception or grabbing is achieved by generating control/guidance strategy based on the desired position of the target in the image plane or using a planning strategy after estimating the target future pose. Target interception using visual information is reported in the literature with techniques such as the Image-Based Visual Servo (IBVS) technique using the target pixel feedback from an image plane for (Kim et al., 2016; Mebarki and Lippiello, 2014), guidance strategy design based on the differential error between the desired and actual target pixel co-ordinates (Mehta et al., 2015; Stepanyan and Hovakimyan, 2007), and developing path-planning strategies after the prediction of target future position and velocities (Cheung et al., 2015; Triharminto et al., 2013). Visual guidance using the concept of Image-Based Visual Servoing (IBVS) for aerial manipulation using UAVs is reported in (Kim et al., 2016; Mebarki and Lippiello, 2014; Lee et al., 2012; Thomas et al., 2014), where the control strategy is developed based on the feedback directly from the image co-ordinates. The position-Based Visual Servoing (PBVS) method is used in another approach, which requires the reconstruction of the target pose with reference to interceptor UAV(Chaumette and Hutchinson, 2006). Vision-based terminal guidance system using a monocular camera is developed using the error between desired and actual pixel co-ordinates (Mehta et al., 2015), where target acceleration is considered as a time-varying disturbance. In (Stepanyan and Hovakimyan, 2007), visual tracking of maneuvering target is proposed using an adaptive disturbance rejection controller where target velocity is considered as a time-varying disturbance. An adaptive guidance strategy is proposed to intercept a maneuvering target in 3D space using LOS angular rate and target image measurements from seeker information (Tian et al., 2011). Generally, it isn't easy to attach a seeker to a multi-rotor UAV system. However, the proposed algorithms in (Mehta et al., 2015; Stepanyan and Hovakimyan, 2007; Tian et al., 2011) are not tested on real images for a real interception scenario.

The interception of a moving target after estimation of the target future position and velocities is reported in (Kumar et al., 2017; Khan et al., 2018). Visual servoing based control strategy is used to capture the target after estimating the target's real time pose using the adaptive extended Kalman filter in (Dong and Zhu, 2016). In (Strydom et al., 2015), Kalman filter-based estimation using stereo vision is employed to intercept moving targets. In (Cheung et al., 2015), target position and velocities are estimated using optical flow, and then an adaptive controller is used to grab the target using the robotic manipulator. Interception of moving target using visual information is proposed in (Zhang et al., 2016), where an online path planning strategy is used to intercept the predicted pose of target considering the target motion model as time series polynomials. In (Jin et al., 2018), motion planning method is used to capture a tumbling satellite using visual information. In (Triharminto et al., 2013), target interception in 3D is developed using dynamic path planning based on the Dubins algorithm. Trajectory optimization technique is used for target interception in (Kritsky et al., 2019; García et al., 2019). Learning-Based Model Predictive Control (LBMPC) is used to catch a ball with quadrotor in an indoor environment (Bouffard et al., 2012). In the above cases, the algorithm is designed based on the pixel error between the desired position and the actual projection of the target on the image plane or involves the target's depth information. Any interception algorithm based on the pixel error/ target depth will be significantly affected by the pixel noise, and it could fail to achieve the terminal accuracy required for grabbing in the outdoor environment. Grabbing an object from a flying target in an uncertain outdoor environment with space, size, and payload restrictions is a far more challenging problem to solve and have not been addressed in the above papers.

Software architecture for mobile multi-robots operation are proposed in several works in the literature such as ALLIANCE-ROS (Li et al., 2016), MAFOSS (Jones et al., 2019), AMEB (Gil et al., 2019), Robosmith (Floroian et al., 2010), (Efremov and Kholod, 2020). Aerial robots pose further constraints due to restrictions in communication bandwidth to satisfy the payload constraints. Various software architectures reported in the literature for multi-agent operation (Dwiyasa et al., 2020; Guerreiro et al., 2019; Sánchez-López et al., 2016; Antonelli et al., 2014) discuss the development of a mission planner for single and multiple robots. Many of these address the aspects of multi-robot handling, but neither contributes an end-to-end solution. Such a solution would comprise algorithms and logic necessary for conflict-free task allocation among multiple agents, with the added features of redundancy and safety. Software frameworks developed for multi UAS operation are reported in the literature, such as AEROSTACK (Sánchez-López et al., 2016), CAVIS (Antonelli et al., 2014); however, a complex mission like Challenge 1 is not demonstrated. In Challenge 1, the mission phases require the allocation of tasks among the agents in a dynamic environment that needs to be carried out over communication with limited bandwidth.

3. Main Tasks and Major Challenges

The main tasks for the successful execution of Challenge 1, adhering to the prescribed constraints, are listed below.

1. Design of an efficient grabbing mechanism while satisfying the volume constraints.
2. Design of a guidance strategy for approaching and grabbing the object from a flying target using visual information.
3. Design of a system architecture for the operation of multiple UAVs.

Since the ball is suspended below the target UAV, close proximity operations are required, which increases the complexity of the task. Apart from the above requirements, we consider the following aspects for designing an overall system.

1. Algorithm requirements
   (a) Algorithms should have real time performance.
   (b) For a reliable mission, the designed algorithms should be robust enough to handle the uncertainty in visual information.
   (c) The uniformly colored target ball lacks rich visual cues and could be indistinguishable from background objects of similar color. The vision algorithm ought to use additional cues like the location of the target UAV and information from multiple frames to get a reliable location of the target ball. In this case, the algorithm should be resilient to variations in shape, size, and appearance of the UAV. Also, as the UAVs are maneuvering at high speeds, the vision algorithm should be capable of detecting the far-off targets rapidly and reliably to allow sufficient time to generate an appropriate response.

   (d) Glare due to lens flare in the outdoor environment can have an impact on color segmentation. Color being one of the few visual features of the target ball, the vision algorithm should do its best to account for visual artifacts in the segmentation process.
2. Hardware considerations
   (a) The manipulator configured at one side of the UAV could avoid a head-on collision with the incoming target, in case of a failed grabbing attempt. An added benefit of the sideways manipulator configuration, which we propose, is the improved redundancy of the system, by deploying the same UAV for both grabbing and balloon-popping in case of any failures. Compared to a manipulator located in the front of the UAV, the sideways design maintains the pitch stability; however, it can reduce the lateral stability. The stability issues that may arise during the mission can be handled by using an adaptive controller.
   (b) Monocular camera, instead of a stereo-camera, could be used to reduce the computational load as well as delay involved in the vision module output. The vision algorithms should be quick and light, which plays a critical role in the interception of flying targets.
   (c) Single degree of freedom (DoF) manipulator is preferable over multi-link manipulators, as in the latter, the computational load for command generation is high and the response is comparatively sluggish for capturing flying targets.
3. Software architecture
   (a) The architecture should be capable of scheduling low-level tasks like inter-agent collision avoidance, obstacle avoidance, etc., as high priority while performing high-level tasks like exploration, tracking, grabbing, etc., for safe multi-UAV missions.
   (b) The task allocation framework should be able to schedule the static and dynamic tasks among the agents dynamically without conflicts.
   (c) The architecture should include basic fail-safe strategies to handle unprecedented situations.
   (d) They should be real time with good computational efficiency.

4. Algorithm Design

To capture an object from a high-speed flying target, algorithms are developed for grabbing, target tracking, estimation, and prediction of the target location. The popping of a balloon involves its detection and an algorithm for stationary target interception using visual information. Accurate path following, inter-agent collision avoidance, and geo-fencing are important features to be included in system design to ensure the safety of multi-UAV operation in close proximity. The controller block should handle the variation in system dynamics over the mission due to wind or change in system parameters. Based on hardware considerations, the conceptual design of the UAV's manipulator system for grabbing and popping is developed as given in Figure 1. The detailed design is described in Section 8. Algorithms are initially developed based on these conceptual designs and later modified based on flight test observations.

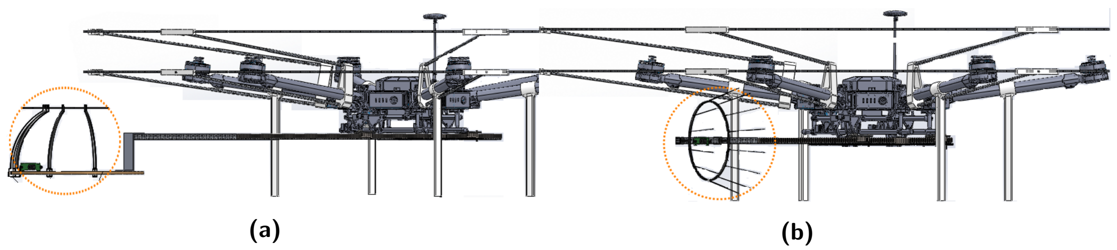


Figure 1. Conceptual designs of (a) Ball-grabbing (b) Balloon-popping manipulators. The proposed sideways mechanisms with their respective end effectors can be seen.

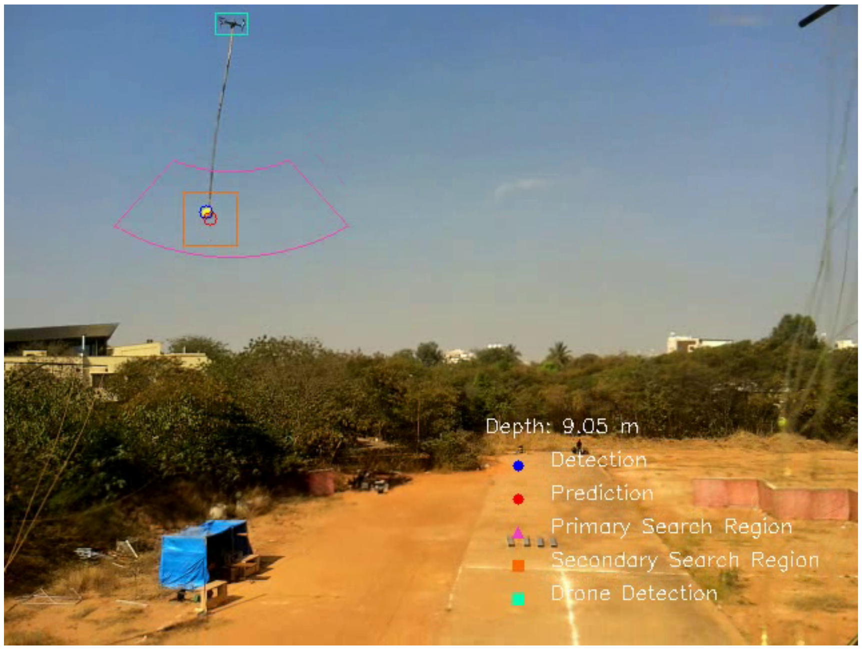


Figure 2. A snapshot showing various regions of interest for the vision module. The color of the ball is yellow.

4.1. Object-detection

This section presents the algorithms for detecting UAV, ball, and balloon. Monocular vision-based method is developed to achieve the detection with good accuracy.

4.1.1. UAV detection

Detecting targets like UAVs flying outdoors can be difficult using traditional techniques due to variations in appearance, caused by sunlight, high velocity of the UAVs, and orientation of the camera. The two-step tracking approach uses TinyYOLOv3 (Redmon and Farhadi, 2018), an object-detection neural network to detect the UAVs and the Kalman filter to track the targets. In the cases of multiple observations, Euclidean distance between the predictions and observations is used by the Hungarian algorithm to associate the observations with the trackers. The tracker uses linear-velocity models to approximate the inter-frame displacement of the targets, and the state of the targets are modeled as:

$$[Cx, Cy, w, h]$$

where, Cx and Cy are x coordinate and y coordinate of the bounding box centre and w and h are width and height of the bounding box, respectively. The algorithm detects UAV as shown in Figure 2.

4.1.2. Ball detection

The vision module for ball detection and tracking involves three parts - Search, Tracking and Grab.

1. **Search Phase:** The main objective of the search phase is to locate the target ball suspended from a moving UAV. The first step is to identify the target UAV. Once the UAV is detected and tracked, its position information is used to search for the ball attached to the target UAV.

   The image is segmented based on the color of the ball, and a vertical region below the bounding box of the UAV in the segmented image is used to search for the target ball. A multivariate Gaussian mixture is used to model the density of the target ball in the images to perform a likelihood-based segmentation (Spurný et al., 2019). We then used the Expectation

Maximization (EM) algorithm to estimate the parameters of the multi-modal distribution from the pixel values of the target ball from training data. In outdoor environments, varying lighting conditions can greatly affect the appearance of the objects and color segmentation in an RGB image whereas HSV color space is robust to lighting conditions as it separates the color information and image intensity information. A reliable segmentation is obtained by using the likelihood of the pixel values from the HSV image. Additionally, sparse optical flow calculated for multiple points inside the segmented contours over successive frames by using the Lucas-Kanade method is used to filter out other distracter contours in the search region. Assuming the target UAV is flying at a higher speed than the UAV with the manipulator, the time of the flow can be used to eliminate some of the false positives. The objects corresponding to the selected contours in the search region are assigned scores based on their geometry and location relative to the corresponding UAV's bounding boxes in the following manner:

$$\text{Circularity} = \frac{4\pi \times \text{Area}}{\text{Perimeter}^2} \tag{1}$$

$$\text{Score} = \begin{cases} \text{Circularity}_{\text{contour}}, & |l_r - \dfrac{L_r \times r_b}{R_b}| \leq \epsilon \\ -1 & \text{otherwise} \end{cases} \tag{2}$$

where, l_r is the distance from the bottom of the UAV to the center of the selected contour and r_b is the radius of the smallest circle enclosing the selected contour; R_b is the actual radius of the target ball, and L_r is the length of the rod used to suspend the ball.

Circularity metric given by (1) is used in the evaluation of the score (2). Ideally, the circularity can range between 0 to 1, with 1 for a perfectly circular contour. However, circularity might be greater than 1 based on the approximations used in measuring the area and perimeter of the contour in digital images. The score is essentially the circularity of the contour, as long as the observed distance of the ball from the UAV in the image is close to the distance predicted. To evaluate the suitability of the contour in consideration for the target ball, the distance of the contour from the UAV in the image l_r is compared with the expected length of the rod in the image for the size of the contour in the image r_b calculated using L_r and R_b; ϵ is the tolerance on the difference between l_r and the expected length of the rod for the images. It is manually tuned by experimentation as the shape of the contours can vary, and the swing of the ball can depend on several factors like the joint connecting UAV and rod, wind conditions, and the motion pattern of the UAV carrying the ball. The scores of the contours corresponding to the detected colored objects are used as a measure of the likelihood of the colored object being the target ball. Once the target ball is identified, perspective projection is employed to compute the position of the target ball using the camera intrinsics and the dimensions of the ball. If a suitable candidate could not be identified, the search for the target ball continues. The block schematic of the search phase is given in Figure 3. The algorithm detects the ball, as shown in Figure 2.

2. **Tracking Phase:** The Kalman filter trackers for the target ball and UAV in the image plane are updated throughout this stage, and the trackers for the other UAVs are removed. The target UAV and ball trackers are used to reduce the region of interest to locate the target ball in the subsequent frames. A wedge-shaped image-region below the UAV is used for this purpose, the location of the wedge from the UAV, and the size of the wedge is determined using the depth information of the target ball from the previous frame. As the dimensions of the ball and the rod used to suspend it are known, the depth information of the target ball from the previous frame is used to estimate roughly the length of the rod in the image plane for the current frame, and this information is used to position the wedge shaped search region accordingly. Similarly, the radius of the target ball from the previous frame is used to determine the size of the wedge-shaped search region.
3. **Grab Phase:** In this terminal phase, the ball must be tracked precisely, without any misses, to grab the ball. As the interceptor UAV approaches the target ball, the target UAV moves

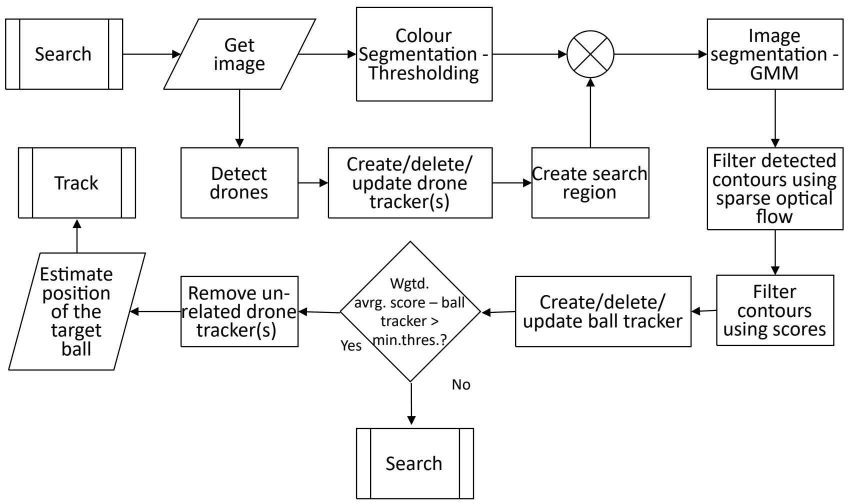

Search
Get image
Colour Segmentation - Thresholding
Image segmentation - GMM
Track
Detect drones
Create/delete/ update drone tracker(s)
Create search region
Filter detected contours using sparse optical flow
Estimate position of the target ball
Remove un-related drone tracker(s)
Wgtd. avrg. score – ball tracker > min.thres.?
Yes
No
Create/delete/ update ball tracker
Filter contours using scores
Search

Figure 3. Schema of search phase in ball detection and tracking.

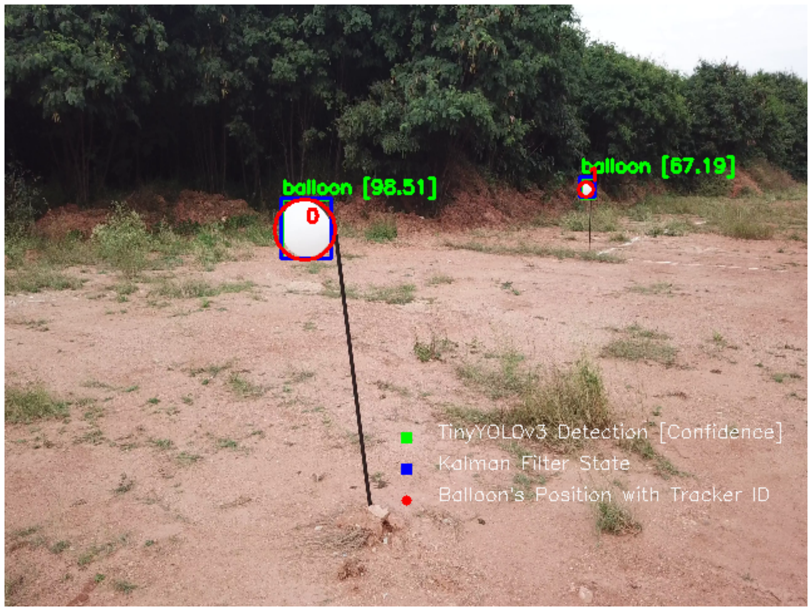

balloon [98.51]
0
balloon [67.19]
TinyYOLOv3 Detection [Confidence]
Kalman Filter State
Balloon's Position with Tracker ID

Figure 4. Balloon detection and tracking. The confidence level of each detection is shown along with the bounding box.

out of the field of view of the camera, and the square-shaped search region around the ball's predicted position is used to track the ball in this phase.

4.1.3. Balloon detection

The balloon-popping UAV detects and tracks the balloons in its field-of-view, as illustrated in Figure 4. Once a balloon is detected, a new instance of the tracker with a unique ID is initialised to monitor the position of the new balloon. Then, the regions of the bounding boxes corresponding to the tracked balloons are segmented based the color of the balloon, and the resulting blobs (potentially representing the balloons) are filtered on circularity. The center and major axis of each blob are computed, and the circle that passes through the intersection points of the major axis and the

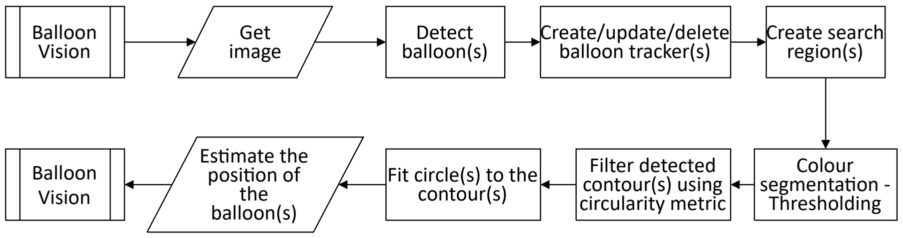

Balloon Vision
Get image
Detect balloon(s)
Create/update/delete balloon tracker(s)
Create search region(s)
Colour segmentation - Thresholding
Filter detected contour(s) using circularity metric
Fit circle(s) to the contour(s)
Estimate the position of the balloon(s)
Balloon Vision

Figure 5. Schematic of balloon detection and tracking algorithm.

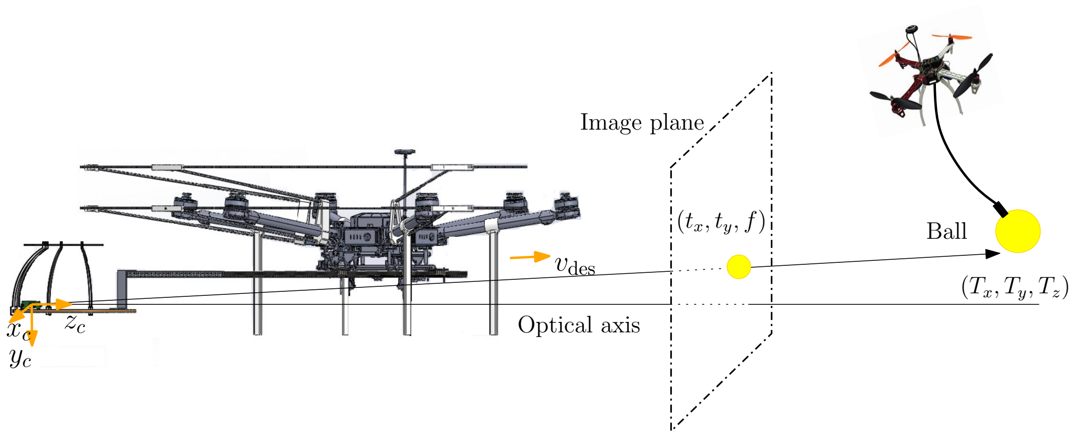


Figure 6. Engagement geometry: The location of target, image plane and manipulator.

contour of the blob is used to estimate the positions of the balloons. This computation assumes that the balloons are spherical in shape, uniform in size, and the diameter is known. The block schematic of balloon detection and tracking is shown in Figure 5. A more detailed account on balloon detection and tracking is given in (Tony et al., 2022).

4.1.4. Use of data sets

We re-purpose TinyYOLOv3 network, trained on PASCAL VOC 2012 dataset as a UAV/balloon detector by fine-tuning its last four convolutional layers. We then use a combination of manually-annotated and synthetically-generated data to re-train the network. A large portion of the dataset is synthetically generated by blending the target images (balloon and UAV images) collected under different lighting conditions and orientations, with diverse backgrounds, sizes, and orientations, as described in (Dwibedi et al., 2017). To extract target images for synthetic data generation, we use Pixel Objectness (Jain et al., 2017), a deep neural network for semantic segmentation, to segment separate targets from background. Also, many negative samples without any annotations are included in the dataset, as this improves the mean average precision for 0.5 IoU threshold (mAP@0.5) by 20% . The inclusion of synthetically generated images in the dataset used for training the network leads to a 20% improvement in mAP@0.9.

4.2. Target tracking

A target tracking algorithm is developed using the information from the monocular camera by combining a vision-based guidance strategy with a distance-based, attractive/repulsive potential function. The guidance strategy keeps the target in the field of view, whereas the attractive/repulsive potential function helps in maintaining the distance from the target. A typical engagement geometry between target and tracker/grabber UAV is shown in Figure 6. In the camera frame, the projected coordinates of the ball in image plane is (t_x, t_y, f). The tracker UAV is made to move parallel to the

Algorithm 1. Ball tracking

```
Input: Tracking distance ($D_{\text{track}}$), target pixel coordinates ($t_x$, $t_y$), camera focal length ($f$), image width ($w$), image height ($h$), yaw attitude ($\psi$), camera frame to body frame transformation matrix ($R_{c2b}$), body frame to vehicle frame transformation matrix ($R_{b2i}$).
 1. Calculate the target pixel-coordinates in the camera frame
    $p_x \leftarrow t_x - w/2, \quad p_y \leftarrow t_y - h/2$
 2. Calculate the distance of ball ($D_{\text{target}}$) from the camera center using the prior information of the size of the ball
 3. Formulate attractive/repulsive potential function considering ($D_{\text{track}}$) as the equilibrium point
 4. Calculate the magnitude of velocity ($V_{\text{track}}$) required from the potential function using the distance from the target
 5. Calculate the unit vector along the line joining the camera center and center of the ball in the image plane, $O^c = (l_x, l_y, l_z)'$
    $l_x \leftarrow \frac{p_x}{\sqrt{p_x^2+p_y^2+f^2}}, \quad l_y \leftarrow \frac{p_y}{\sqrt{p_x^2+p_y^2+f^2}}, \quad l_z \leftarrow \frac{f}{\sqrt{p_x^2+p_y^2+f^2}}$
 6. Calculate the desired velocity in the camera frame
      if $D_{\text{target}} \geq D_{\text{track}}$ then
        $V_c \leftarrow V_{\text{track}} O^c$
      else
        $V_c \leftarrow -V_{\text{track}} O^c$
      end if
 7. Calculate the transformation from the camera frame to the vehicle frame: $R_{c2i} \leftarrow R_{b2i} R_{c2b}$
 8. Desired velocity in the vehicle frame: $V_{\text{des}} \leftarrow R_{c2i} V_c$
 9. Calculate the $O^c$ in the vehicle frame: $O^v \leftarrow R_{c2i}\, O^c$
10. Calculate desired yaw: $\psi_{\text{des}} \leftarrow \arctan(\frac{O^v(2)}{O^v(1)})$
11. Calculate yaw offset: $e_\psi \leftarrow \psi_{\text{des}} - \psi$.
12. Calculate desired yaw rate: $r_{\text{des}} \leftarrow k p_\psi e_\psi + k d_\psi \frac{\mathrm{d}e_\psi}{\mathrm{d}t}$
Output: Desired velocity ($V_{\text{des}}$), desired yaw rate ($r_{\text{des}}$).
```

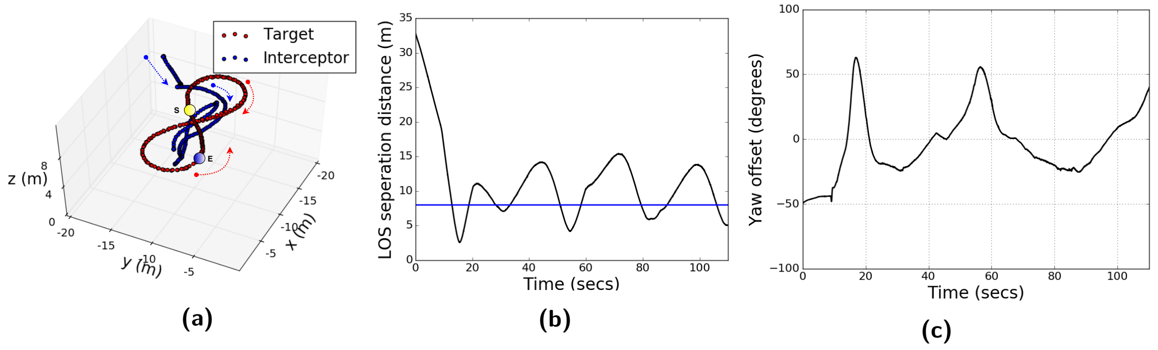


Figure 7. Target tracking (a) Trajectory (b) Distance between the target and interceptor (c) Yaw offset during tracking.

line joining the camera focal point and the pixel position of the target's center in the image plane, so that it can follow the target point. If the UAV is inside the tracking radius of the target, then the UAV moves away from the target and *vice-versa*. Apart from the commanded velocities, a yaw rate command is also used to maintain target within the field of view of the camera attached to the manipulator. Algorithm 1 shows the steps of the tracking process. A tracking scenario is simulated in the Gazebo to track a target moving in a figure-of-eight loop, where the tracking radius is 8 m. Figure 7a shows the trajectories of the target and the tracking UAV. The separation between the target and tracking UAV is plotted in Figure 7b which shows that the tracking UAV remains 8 m away from the target UAV. The oscillatory response is due to the tracker UAV being pushed away

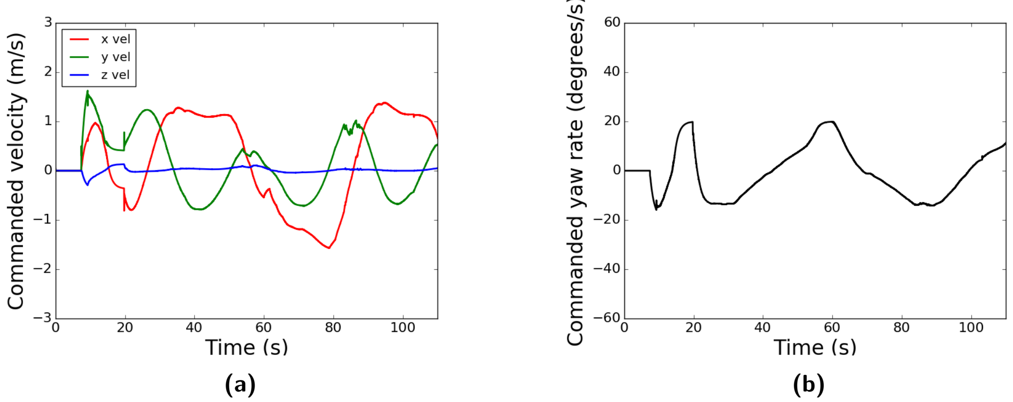


Figure 8. (a) Commanded velocity in x, y and z direction (b) Commanded yaw rate.

by the potential field force when closing in to the target. Figure 7c shows the error between desired yaw and actual yaw angle. The commanded values of position and heading are shown in Figure 8.

4.2.1. *Computational complexity*

Consider that ith step in the algorithm takes T_i units of time to execute. There are 12 steps in this algorithm. The total time involved is the summation of individual steps involved.

$$T_{\text{Total}} = T_1 + T_2 + T_3 + \cdots + T_{12} \tag{3}$$

where, T_{Total} is the total execution time of the algorithm. Thus, for a given camera and its parameters, object-detection algorithm, and vehicle state parameters, the run-time computational complexity is a constant. At every time step, a constant run time is required for tracking the ball, which makes it feasible to implement in real time.

4.3. Guidance for ball-grabbing/balloon-popping

In this section, a guidance strategy is designed for ball-grabbing and balloon-popping. The method is inspired by the missile guidance literature, where target interception is the focus. The design of the end-effector of the manipulator is also arrived at, based on the guidance strategy.

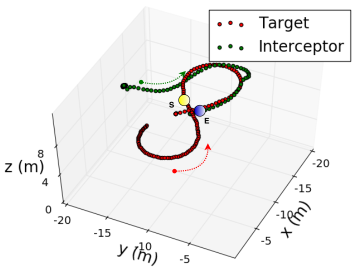
4.3.1. *Guidance for moving ball-grabbing*

Various guidance strategies such as different forms of Proportional Navigation (PN) are reported in the literature for interception of a target using radar/seeker information (Zarchan, 2012), (Siouris, 2004). However, it is difficult to extend the same algorithms using visual sensors because of the delay and uncertainty involved in its sensor data. In missile interception problems, the expected miss distance in the terminal phase is accommodated by considering the suitable lethal radius of the attached warhead. In our case, the allowable miss distance for balloon-popping or ball-grabbing is very low, considering the size of the manipulator to satisfy the overall size constraints. Any guidance strategy involving the direct or indirect target depth information will have a large miss distance because of the uncertainty involved in the estimation of depth. The miss distance will further increase if the depth information is obtained from a monocular camera. In the case of the interception of a flying target, predictive guidance strategies will also have large miss distance due to uncertainty involved in predicted target position from the visual sensor data.

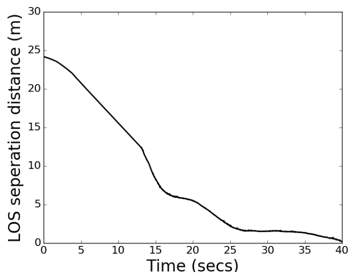
Considering the above facts, a visual guidance strategy is developed based on the idea of the classical pure pursuit guidance, where the interceptor moves towards the target along the line of sight between the camera center and ball center without including the target depth information. As shown in Figure 6, the grabber UAV applies velocity parallel to the line joining the camera centre

Algorithm 2. Ball-grabbing

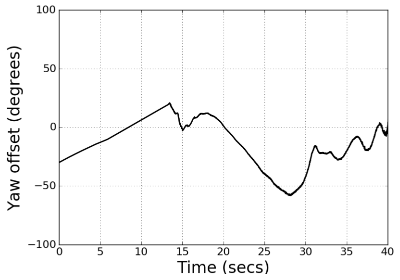
```
Input: Target pixel coordinates ($t_x$, $t_y$), camera focal length ($f$), magnitude of target velocity ($V_t$), excess velocity ($V_{\text{excess}}$), image width ($w$), image height ($h$), yaw attitude ($\psi$), camera frame to body frame transformation matrix ($R_{c2b}$), body frame to vehicle frame transformation matrix ($R_{b2i}$)).
1. Calculate the target pixel-coordinates in the camera frame
   $p_x \leftarrow t_x - w/2, \quad p_y \leftarrow t_y - h/2$
2. Calculate the unit vector along the line joining the camera center and center of ball in image plane
   $O^c \leftarrow \frac{1}{\sqrt{p_x^2+p_y^2+f^2}}(p_x, p_y, f)$
3. Desired velocity magnitude for interception in camera frame: $V_{\text{mag}} \leftarrow V_t + V_{\text{excess}}$
4. Desired velocity for interception in camera frame: $V_c \leftarrow V_{\text{mag}} O^c$
5. Calculate the desired velocity components in camera frame: $V_c = (V_{cx}, V_{cy}, V_{cz})'$
   $V_{cx} \leftarrow V \frac{p_x}{\sqrt{p_x^2+p_y^2+f^2}}, \quad V_{cy} \leftarrow V \frac{p_y}{\sqrt{p_x^2+p_y^2+f^2}}, \quad V_{cz} \leftarrow V \frac{f}{\sqrt{p_x^2+p_y^2+f^2}}$
6. Calculate the transformation from camera frame to vehicle frame: $R_{c2i} \leftarrow R_{b2i} R_{c2b}$
7. Desired velocity in vehicle frame: $V_{\text{des}} \leftarrow R_{c2i} V_c$
8. Calculate the desired yaw rate $r_{\text{des}}$ as described in Algorithm 1
Output: Desired velocity ($V_{\text{des}}$), desired yaw rate ($r_{\text{des}}$).
```



(a)



(b)



(c)

Figure 9. (a) UAV trajectories (b) Variation of LOS separation (c) Yaw offset during grabbing. The LOS distance converges to zero on grabbing.

and the projection of centre of ball (T_x, T_y, T_z) on the image plane (t_x, t_y, f). The grabber UAV also applies yaw rate command to keep the ball in the center of the field of view of the camera. The details of the guidance algorithm are presented in Algorithm 2. Since the target is moving at low speed, the turn radius required at the last phase of grabbing is within the allowable limit of the grabber UAV.

A typical engagement scenario is generated to validate the grabbing algorithm in Gazebo. Figure 9a shows the target and interceptor path for an engagement scenario where, the target is detected for the first time at point S, and the engagement ends at point E. Target and interceptor speeds are 2.0 m/s and 2.5 m/s, respectively. The variation of Line of Sight (LOS) separation distance between the camera center and the target during the grabbing process is plotted in Figure 9b. Clearly, the distance goes to zero during grabbing. The error in the desired yaw to keep the target in the center of the field of view and the actual yaw is shown in Figure 9c. The commanded values of position and heading are shown in Figure 10.

The effective radius of the gripper of the grabbing drone's manipulator is designed based on the form of the guidance strategy. It was observed during the initial flight tests that the grabber UAV was unable to perform quick maneuvers when the ball depth was less than 0.5 m. This was attributed to the computational delay associated with the vision module. To resolve this, the algorithm for ball-grabbing was designed so that the end effector of the grabber UAV reached feasible relative positions for successful capture before it reached 0.5 m to the ball. An analysis of relative velocity

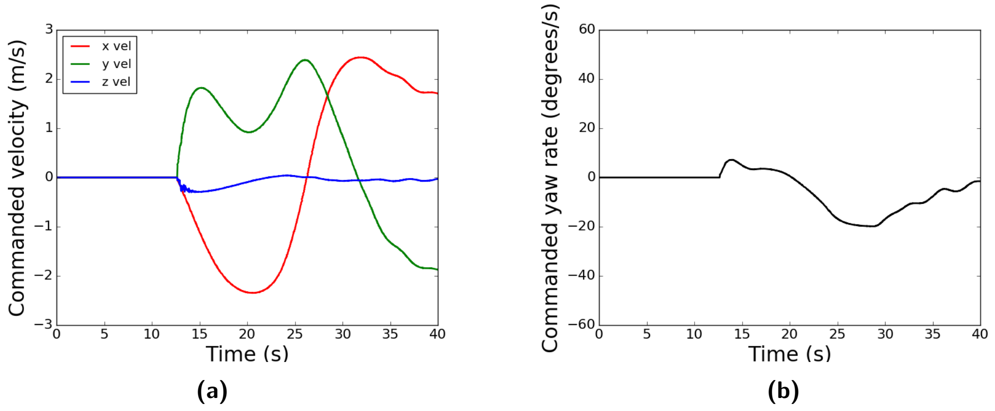


Figure 10. (a) Commanded velocity in *x*, *y* and *z* direction (b) Commanded yaw rate.

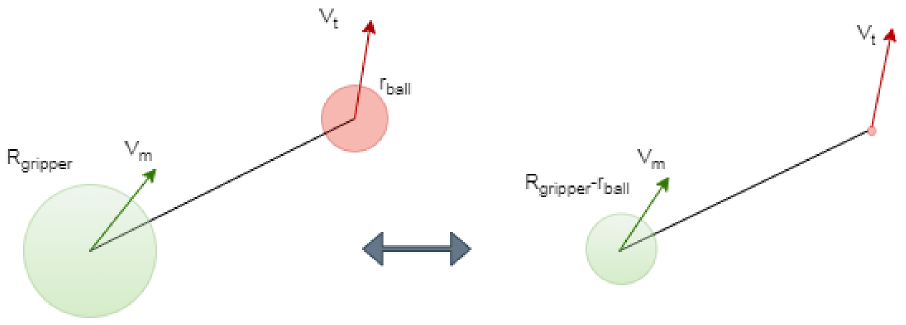


Figure 11. Ball grabber end-effector design: The virtual sphere is constructed to compute the grabber size.

space between the target and the UAV was performed considering the target velocity constant at the terminal phase.

Consider an engagement scenario, as shown in Figure 11, where the ball with radius r is moving with constant velocity V_t and a gripper having effective grabbing radius of R_{gripper}, moves at a speed of V_m, pursuing the ball. The grabbing can be expressed as an equivalent scenario of interception of a virtual sphere with radius R_{eq}, where the target is assumed to be a point. For successful grabbing of the ball, the equivalent grabbing radius is the difference between the gripper's radius and the radius of the ball.

$$R_{eq} = R_{\text{gripper}} - R_{\text{ball}} \tag{4}$$

Interception is possible if the point target lies in the collision cone of the virtual sphere (Chakravarthy and Ghose, 2012). Therefore, the radius of virtual sphere should satisfy the following condition.

$$R_{eq}^2 \geq \frac{r_0^2(V_{\theta 0}^2 + V_{\phi 0}^2)}{(V_{\theta 0}^2 + V_{\phi 0}^2 + V_{r0}^2)} \tag{5}$$

where, r_0 is the initial distance, V_{r0} is the component of the relative velocity along the line of sight, $V_{\theta 0}$, $V_{\phi 0}$ are the relative velocity components normal to the line of sight. When the ball is 0.5 m away from the end-effector, the guidance strategy should position the grabber UAV such that the error in alignment of target velocity and interceptor velocity is within 5°. Considering an approximate pure pursuit engagement in the 2D plane, a head-on scenario is considered with a target speed of 8 m/s and an interceptor speed of 2 m/s. The value of R_{eq} should be greater than 34.9 mm, as obtained from the (5), where, the values of r_0, V_{r0}, $V_{\theta 0}$, $V_{\phi 0}$ are 0.5 m, 9.96 m/s, 0.69 m/s and 0 m/s, respectively. A similar analysis for a tail-chase condition with a target speed of 4 m/s and

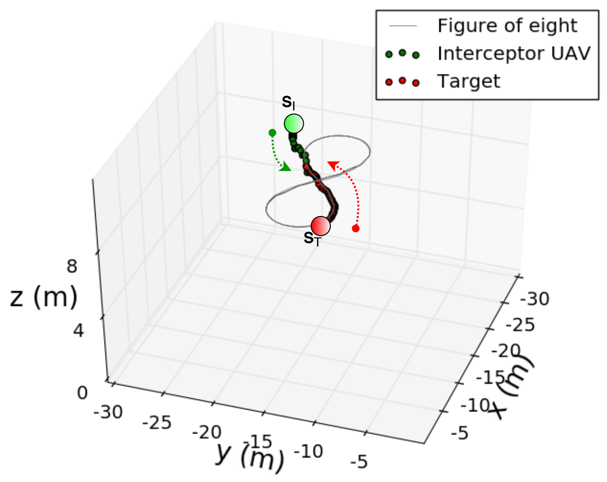


Figure 12. Engagement with high speed target.

interceptor speed of 6 m/s gives the minimum value of R_{eq} to be 85 mm. The size of the target ball is 100 mm. As per (4), the effective radius of the gripper should be greater than 185 mm. Thus, the design value of the effective radius of the end-effector is chosen as 200 mm.

4.3.2. Computational complexity

We follow the same procedure as in the previous section to evaluate the run time complexity of the ball-grabbing algorithm. The total run time of the algorithm is computed as below.

$$T_{\text{Total}} = T_1 + T_2 + \cdots + T_7 + T_8 \tag{6}$$

Step 8 constitutes three computations. Thus, the run-time complexity of ball-grabbing algorithm is also a constant, making it a fit candidate for real-time implementation.

4.3.3. Interception of high speed target

The interception of a high-speed target requires the prediction of the target points with reasonable accuracy. In our case, it is known a priori that the target will be moving through a repetitive trajectory and a ball with known color will be attached to the target at a certain distance below the target. Using these information of the type of target path and the color of the object, an estimation framework using EKF is developed to estimate the position of the target. The information of the repetitive trajectory of the target is considered in the motion model, and the measurement model is developed using the visual observations of the target points. After the estimation of target points over a duration, the future position of the target UAV is predicted from the previous sequence of observations. As the target UAV is performing a figure-of-eight loop, a curve fitting method is used to estimate its trajectories based on the observed points. After estimation of the target path, the suitable grabbing stand-off points are selected such that the grabber UAV can perform approximate head-on interception with the target. Grabbing stand-off points are those feasible locations with respect to the target trajectory where, the UAVs should wait to capture the ball. A typical interception scenario of high speed target is shown in Figure 12. The grabber UAV waits at the stand-off points and follows the target to capture the ball. A detailed account of the estimation and prediction framework is given in (Bhise et al., 2020; Agrawal et al., 2020).

4.3.4. Computational complexity

The total time for execution of Algorithm 3 is given below.

$$T_{\text{Total}} = T_{1a} + T_{1b} + T_{1c} + T_{1d} + T_2 + T_3 + T_4 + T_5 + T_6 \tag{7}$$

Algorithm 3. Target position estimation, prediction and grabbing stand-off point estimation

```
Input: Target pixel coordinates in camera frame ($p_x$, $p_y$), focal length ($f$), target depth ($Z$), magnitude of target velocity ($V_T$).
    1. Develop a EKF using the target position as state vectors
        (a) Calculate the approximate centre of curvature ($T_{x0}$, $T_{y0}$) of path of target
        (b) Consider the state dynamics:
            $\dot{T}_x \leftarrow -V_T \frac{T_x - T_{x0}}{\sqrt{(T_x - T_{x0})^2 + (T_y - T_{y0})^2}}$,   $\dot{T}_y \leftarrow -V_T \frac{T_y - T_{y0}}{\sqrt{(T_x - T_{x0})^2 + (T_y - T_{y0})^2}}$
        (c) Measure the states vectors using the target centre pixel co-ordinates and target depth ($Z$):
            $T_x \leftarrow \frac{p_x Z}{f}$,   $T_y \leftarrow \frac{p_y Z}{f}$
        (d) Estimate the current position of target ($\hat{T}_x$, $\hat{T}_y$) with the EKF formulation
    2. Predict the incremental movement of target ($\delta_{T_x}$, $\delta_{T_y}$) at future $m$ steps from the previous $n$ sequence of observations
    3. Predict the target location ($\hat{T}_{mx}$, $\hat{T}_{mx}$): $\hat{T}_{mx} = \hat{T}_x + \delta_{Tx}$,   $\hat{T}_{my} = \hat{T}_y + \delta_{Ty}$
    4. Estimate the average height of target ($\hat{T}_z$)
    5. Estimate the trajectory of the path by fitting best curve after minimizing the least square errors
    6. Estimate suitable stand-off point ($S_x$, $S_y$, $S_z$) near to target path for grabbing
Output: Target current position ($\hat{T}_x$, $\hat{T}_y$, $\hat{T}_z$), predicted position ($\hat{T}_{mx}$, $\hat{T}_{my}$, $\hat{T}_z$), grabbing stand-off point ($S_x$, $S_y$, $S_z$).
```

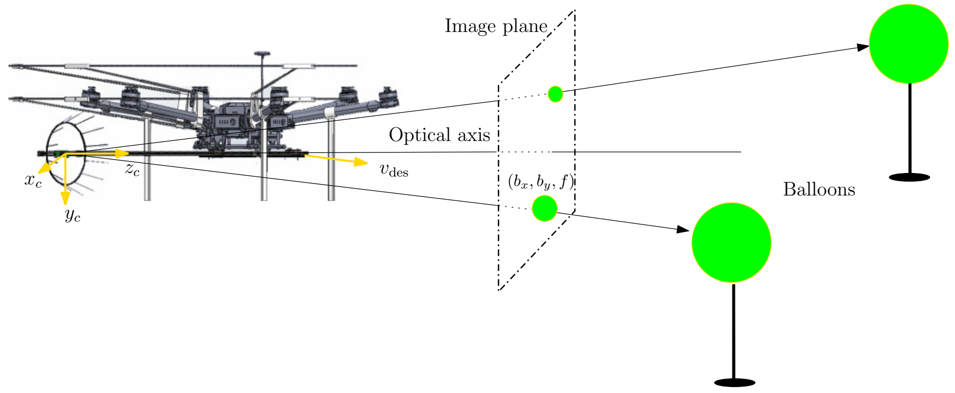


Figure 13. Balloon-popping: the geometry showing the relative position of camera, image plane and the target balloons. A nearer balloon is approached first upon detecting multiple balloons.

where, T_{1a}, T_{1b}, T_{1c}, T_{1d} denote the time involved in the steps 1a-1d, respectively. Let the time associated to compute the estimation algorithm be T_{est}. Steps 1d involves estimation of states. Estimation involves computing the covariance matrix, Kalman gain, and the estimated states using the state equations, and the previous measurements. For step 2, it is required to compute predictions for m time steps and hence, the run-time computational complexity of the algorithm is dependent on the number of prediction steps added to a constant value for the other computations, that is, $O(\mathrm{m})$ + constant. Depending on the computational capability and the accuracy required, the designer can decide on a trade-off between the two considerations.

4.3.5. Balloon interception

The guidance strategy developed for ball-grabbing is modified to intercept the balloon, considering it as a stationary target. The engagement geometry during a balloon-popping scenario is shown in Figure 13. The UAV pops the nearest balloon first if multiple balloons are detected. In case of balloon interceptor UAV, the end effector is designed such that a nominal contact is sufficient to achieve successful popping. So, the equivalent radius desired for popping is the sum of the radius of

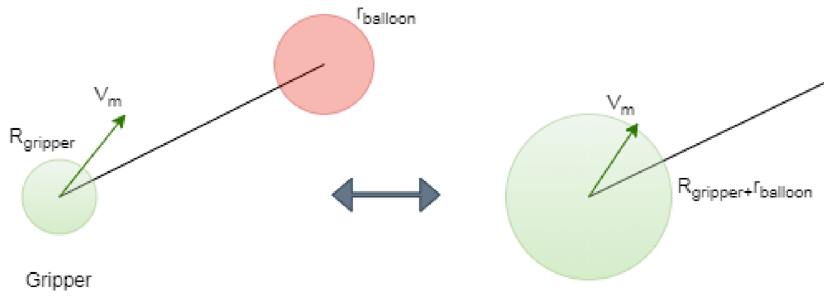


Figure 14. Balloon popper design. A comparatively small gripper is obtained for balloon interception.

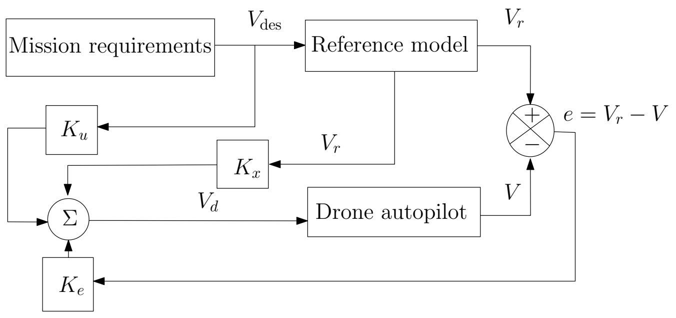


Figure 15. Controller block diagram. This simple adaptive controller handles the possible uncertainties.

the gripper and the radius of the balloon (when fit into the resultant virtual sphere, Figure 14).

$$R_{eq} = R_{\text{gripper}} + R_{\text{balloon}} \tag{8}$$

An analysis similar to that of grabbing drone gives the minimum virtual radius sphere to be 211 mm for the balloon popper speed of 2 m/s and alignment error of 25° at a distance of 0.5 m. The alignment error margin is kept high as the time of engagement for balloon popper UAV is less. Considering the balloon radius of 150 mm, the effective gripper radius of the balloon popper UAV is found to be 61 mm (from (8)). The final value of the effective radius of balloon popper UAV is chosen as 100 mm.

4.4. Controller design

A variation in system parameters is observed during and after the grabbing phase compared to other task phases. Also, disturbances are expected due to wind gust. So, the controller needs to adapt to these variations. We have designed an add-on adaptive controller based on the concepts of Simple Adaptive Controller (SAC) to handle the uncertainty. The desired velocity generated from the task module is modified to accommodate the uncertainty. A typical block diagram is shown in Figure 15, where the desired velocity command from the task module (V_{des}) is modified using add-on adaptive controller before feeding to the UAV autopilot. In Figure 15, V_d is the commanded velocity to the autopilot, V is the actual velocity, V_r is the velocity output from the reference model and K_x, K_e, K_u are the controller gains. Separate controller structure for velocity components and the yaw rate is developed and described in Algorithm 4. Following the previous method to compute the worst-case run time, the computational complexity of the controller algorithm is also a constant.

4.5. Geo-fencing

The UAVs should remain within the arena bounds while executing its tasks. A robust geo-fencing algorithm is needed to ensure that the UAVs remain within the predefined physical bounds. The

Algorithm 4. Controller Design

```
Input: Desired velocities ($V_{\text{desx}}, V_{\text{desy}}, V_{\text{desz}}$), desired yaw rate ($r_{\text{des}}$), velocities ($V_x, V_y, V_z$), yaw rate ($r$), gains ($k_{ex}, k_{ey}, k_{ez}, k_{er}, k_{xx}, k_{xy}, k_{xz}, k_{xr}, k_{ux}, k_{uy}, k_{uz}, k_{ur}$).
  1. Select individual reference models for the desired responses of actual velocities and yaw rates against the commanded values
  2. Calculate the output of reference models: $V_{rx}, V_{ry}, V_{rz}, r_r$
  3. Calculate the error between the reference model output and the actual value:
     $e_x \leftarrow V_{rx} - V_x$
     $e_y \leftarrow V_{ry} - V_y$
     $e_z \leftarrow V_{rz} - V_z$
     $e_r \leftarrow r_r - r$
  4. Commanded velocity to autopilot:
     $V_{dx} \leftarrow k_{ex} e_x + k_{xx} V_{rx} + k_{ux} V_{\text{desx}}$
     $V_{dy} \leftarrow k_{ey} e_y + k_{xy} V_{ry} + k_{uy} V_{\text{desy}}$
     $V_{dz} \leftarrow k_{ez} e_z + k_{xz} V_{rz} + k_{uz} V_{\text{desz}}$
  5. Commanded yaw rate to autopilot:
     $r_d \leftarrow K_{er} e_r + k_{xr} r_r + k_{ur} r_{\text{des}}$
Output: Commanded velocities ($V_{dx}, V_{dy}, V_{dz}$), commanded yaw rate ($r_d$).
```

arena is 100 x 60 x 20 m and is covered with mesh. The boundaries of the environment are set on a global frame. So, the convex 3D hull is chosen as a candidate geometric representation of the geofence. Since the boundaries of the arena are known, we use the 3D quick hull algorithm (Barber et al., 1996) to compute the set of faces, which is a set of vertices, corresponding to the face created by the boundary points. The 3D quick hull is an iterative algorithm that adds individual points one after the other to create intermediate hulls. It creates an initial hull with faces and iteratively visits all the faces to find the horizon. After termination, a list of all the horizons is obtained in a counter-clockwise direction, which are the edges of the convex hull. The faces of the physical boundaries are obtained, and the nearest point projected by the UAV on each face is calculated. These projected points are treated as obstacles for the UAV, and at a certain distance, the UAV is subjected to a repulsive force along the line of sight of the projected point and the UAV. A commanded velocity keeps the UAVs away from the arena boundary.

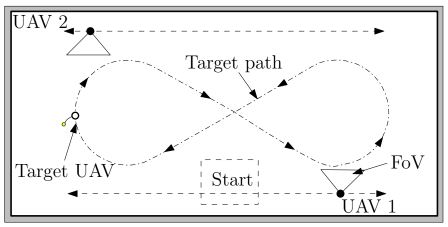
4.6. Collision avoidance

To avoid unforeseen inter-UAV collisions, an artificial potential field algorithm is designed based on collision cone criteria (Chakravarthy and Ghose, 2012). Collision avoidance based on proportional navigation using the visual sensor data is reported in (Clark et al., 2015). However, capturing and processing the visual information from all sides is not feasible considering the space, power, and payload constraints of the UAV. The collision avoidance algorithm is designed by sharing the position and velocity information among the UAVs. The relative velocity along the line of sight (V_{r0}) and its normal (V_{θ_0}, V_{ϕ_0}) are used to check the collision condition. If the distance between the pair of UAVs is r_0 and the radius of the safe sphere to avoid the collision is R, then the following condition is used to check if they are in the collision cone (Chakravarthy and Ghose, 2012).

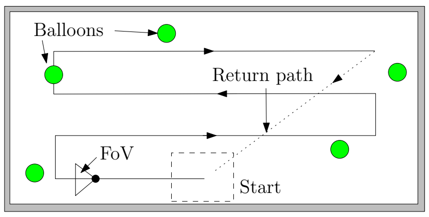
$$r_0^2(V_{\theta 0}^2 + V_{\phi 0}^2) \le R^2(V_{\theta 0}^2 + V_{\phi 0}^2 + V_{r0}^2) \tag{9}$$

$$V_{r0} < 0 \tag{10}$$

Once a UAV is in the collision cone and within a certain distance from an incoming UAV, it is subjected to acceleration away from the UAV along the line of sight of both UAVs. The magnitude of the acceleration is selected based on the velocity and initial distance between the UAVs. In some cases, one UAV is assigned high priority if it is involved in carrying out any critical tasks and the other UAVs as low priority. In those cases, only the low priority UAV maneuvers to avoid the collision.



(a)



(b)

Figure 16. Exploration pattern for (a) grabbing (b) popping. The UAVs face the camera to the inside of the arena while exploring to detect the ball, while for exploration to detect balloons, the UAV heading is along the path.

5. Mission Modes

The overall mission is classified into several tasks so that it can be allocated to different UAVs efficiently. The tasks are designed to minimize the conflicts among the them as well as to expedite the overall completion time of the mission. The important mission modes or tasks are take-off, exploration, tracking, search and grabbing, popping and landing. They are described below.

5.1. Exploration

The exploration task is accomplished through waypoint-based navigation wherein a set of waypoints are supplied to the UAVs. Grid-based waypoints, in an exhaustive lawn-mower pattern, are fed to the UAV performing the balloon-popping task. On the other hand, waypoints with a horizontal scanning pattern between two locations are fed to the UAVs performing the ball-grabbing task. The UAVs, while navigating through waypoints, orient themselves in a direction that keeps the camera field of view perpendicular to the waypoint trajectory and toward the inside of the arena, as shown in Figure 16. The UAV performing the balloon-popping task, however, always orients towards its next waypoint.

5.2. Target tracking

The purpose of the target tracking task is to provide the position data of the path followed by the target. This position data is used to estimate the type and parameters of the repetitive curve followed by the target. The target tracking task is triggered if a UAV detects the target while performing the exploration task. This task is considered complete once the curvature parameters of the target UAV's path are estimated. Only one UAV performs this task while the other UAV stays at a standby position. The standby position is fed into the algorithm before execution, considering the feasibility of detection and grabbing within the given arena. Upon losing the target while tracking, the UAV goes to a standby position and the other UAV resumes tracking. Thus, the tracking is performed in a cooperative fashion.

5.3. Grabbing

The grabbing task is two-fold. The interceptor UAV should be guided to detach the ball, and once it is detached, feedback needs to be generated. Thus, the sub-tasks associated with the grabbing operation are the following.

1. **Aerial manipulation:** While performing ball-grabbing operation, the UAV guides the camera attached to the manipulator to the center of the ball. While doing so, the rigid part of the manipulator must hit the magnetic link attached to the target ball. During the terminal phase, when the ball fully covers the field-of-view of the camera, additional forward momentum is

generated by the UAV so that it can detach the magnetic link with the detachment mechanism of the manipulator. The detached ball drops into the manipulator net.

2. **Grab detection:** On the bottom of the manipulator end effector, limit switches are placed over a circular mount. They are calibrated to activate when the ball falls on them. The calibration is such that a minimum weight of 50 g is enough to detect the ball's presence in the basket. The triggering of limit switches sends the desired feedback and completes the grabbing task. The mesh is fixed to the end-effector in a conical shape to ensure the falling of the ball over the limit switches.

The UAVs cooperate to achieve successful grabbing. In case a UAV loses the ball from its FoV, the other UAV resumes its grabbing task while the former moves to the grabbing stand-by location. This not only improves the probability of grabbing but also makes the grabbing operation quick and efficient. A detailed account on UAV collaboration for moving target capture is given in (Tony et al., 2021; Tony et al., 2020).

5.4. Balloon-popping

This task starts when a balloon is detected during the balloon exploration. The UAV registers its position at which the detection occurred and estimates the position of the detected balloon. This helps in the confirmation of popping later. The UAV proceeds to pop the balloon, by servoing towards the tracked center of the balloon in the FoV. Forward velocity commands are generated until there is no balloon in the camera field of view for a few continuous frames, and the current position of the UAV is beyond the position of the balloon estimated earlier. To determine the relative position of the UAV with respect to the balloon, two vectors, originating from the current UAV position and one ending at the earlier registered UAV position and the other at the balloon's estimated position, are used to compute a dot product. The sign change of this vector determines if the UAV has gone beyond the balloon's position. Further, to confirm a successful pop, the UAV returns to the previously registered position and checks if there is a balloon in its field of view. If not, the UAV gets back to the exploration mode and continues search for balloon. The program also keeps count of the popped balloons. The task is assumed to be complete when the number of popped balloons is equal to five or if no balloon is detected in two rounds of exploration. A flow chart for the balloon-popping module is shown in Figure 17. It summarises the overall flow of control while the balloon-popping module is being executed.

5.5. Other subsidiary modules

Apart from the above modules, the two other essential modules are the take-off and landing modules. The take-off module is run through a standard UAV take-off Software Development Kit (SDK) routine, whereas landing is performed using assistance from vision and a landing SDK routine.

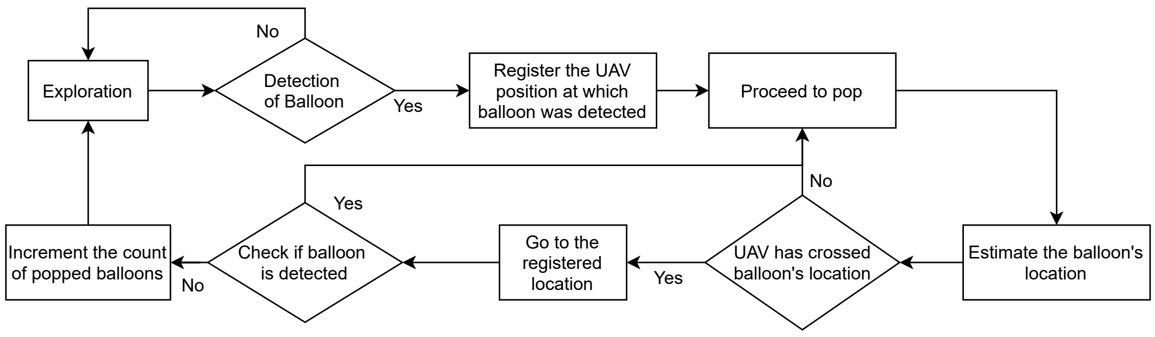

Exploration
No
Detection of Balloon
Yes
Register the UAV position at which balloon was detected
Proceed to pop
Yes
No
Increment the count of popped balloons
No
Check if balloon is detected
Go to the registered location
Yes
UAV has crossed balloon's location
Estimate the balloon's location

Figure 17. Balloon-popping task.

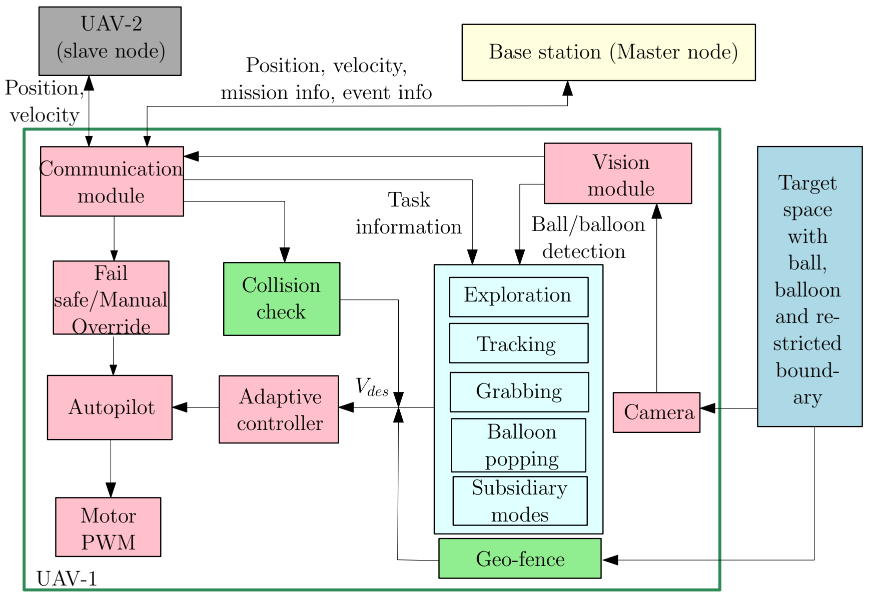


Figure 18. The OMS architecture and flow of control among UAVs and the base station.

Another module is grabbing standby, where the UAV moves to the standby location and waits until it receives the command to switch to grabbing. Detection of the ball, balloon, and landing zone is performed through different cameras. Detection modules are run in parallel to the main task node so that visual sensor data is obtained continuously. To handle resets in the competition, a restart module is also included along with the subsidiary modes. This will ensure that all the mission parameters are saved for resuming the mission after the reset is executed. This task essentially acts as a 'pause' for the mission execution.

6. Operation Management System (OMS)

We developed a complete software architecture in ROS for autonomously deploying multiple UAVs to carry out the challenge. Two UAVs collaborate to grab the ball while the third one is dedicated to pop the balloons. The software architecture handles the flow of control to execute the entire mission. It is also responsible for coordinating the UAVs to execute the operation smoothly. The OMS coordinates task allocation amongst the UAVs by sharing the task information and their state information through a multi-master network setup. The OMS architecture is shown in Figure 18. It shows how the different mission modules integrate within the system. The mission modes coordinate with other system modules and with the base station. Specific aspects like collision avoidance are achieved by communication with the other UAVs.

In a multi-master network, the multi cast feature of network adapters is used to share the topics amongst the different nodes on different computers over the network. The IP addresses of all the computers in the network are added to the multi cast group in each local computer. This allows for inter-UAV communication of the task and agents state information among the agents in the multi cast group. In this setup, the master node does task allocation and provides the task information to the slave nodes based on the current states of all agents and mission requirements. Master nodes receive important state information such as position and velocity, along with the status of the current task of the slave nodes. Slave nodes share only the position and velocity information among themselves to avoid collision among themselves. The master node can be on the base station or on any of the UAVs. However, keeping the master node in the base station is preferable, considering the computational requirement for the master node as well as to improve the safety of the overall

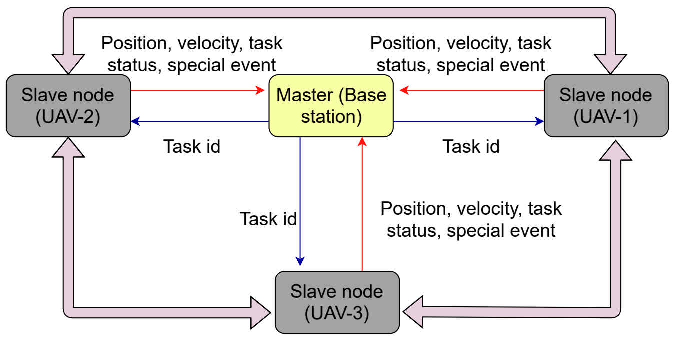

Position, velocity, task status, special event
Position, velocity, task status, special event
Slave node (UAV-2)
Master (Base station)
Slave node (UAV-1)
Task id
Task id
Task id
Position, velocity, task status, special event
Slave node (UAV-3)

Figure 19. Information sharing in the base station-UAV network.

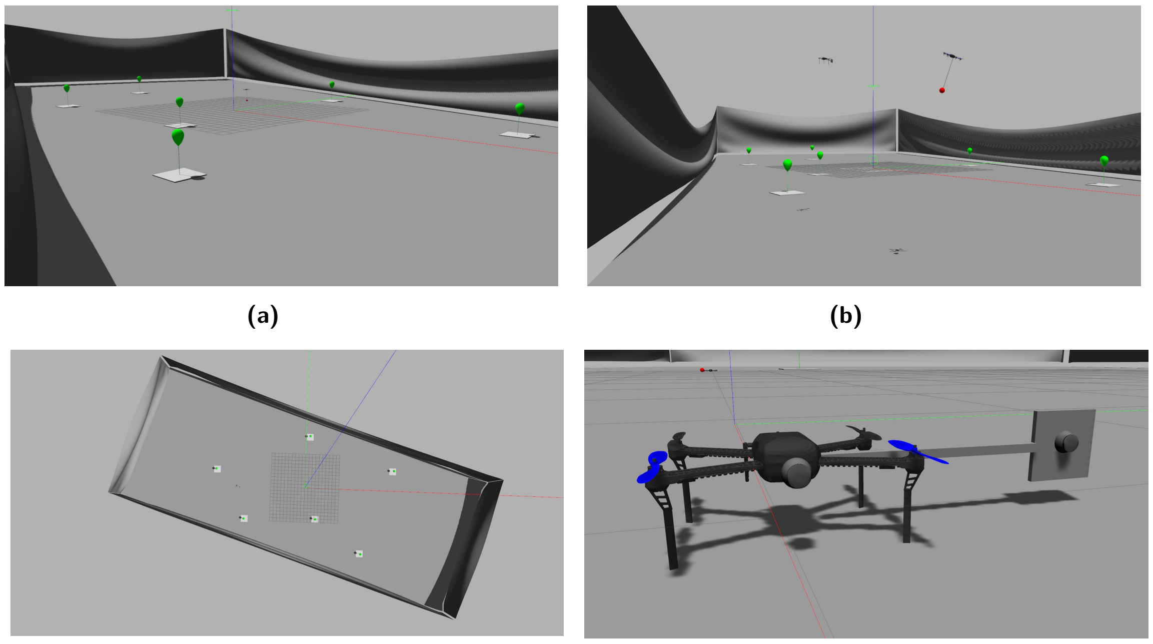
system. The information sharing among the different agents of the network is kept at a minimum to avoid the load on the network. The schematic representation of the communication system is shown in Figure 19. Besides managing the tasks, OMS also does some basic functionalities. These include task allocation, reallocation in case of failure, fail-safe mechanism and system monitoring.

1. **Task allocation:** OMS schedules the tasks requiring the control inputs; whereas, tasks like capturing images and ball/balloon detection are run as a separate module continuously. The tasks are classified based on priority. Safety-critical tasks like inter-agent collision avoidance, geo-fencing, etc. are given high priority; whereas, mission-specific tasks like exploration, tracking, and grabbing are given low priority. The high priority tasks are run in parallel while only one mission specific task is selected at a time for execution. OMS selects the mission specific tasks based on the state of the agents and mission requirement. The desired acceleration from the tasks and the safety-critical tasks are combined to obtain the commanded velocities for the current mission.

   The mission specific tasks are further classified into static tasks and dynamic tasks. Static tasks are predefined for a mission, whereas dynamic tasks are event-triggered tasks. These triggering events can be the detection of a target, grabbing of the ball, or popping of a balloon, etc. Some examples of static tasks are exploration for balloon and ball detection, whereas dynamic tasks are target tracking triggered by the detection of the ball by the UAV. OMS can create or delete tasks as per the mission requirement.
2. **Task reallocation and redundancy:** Task reallocation is useful if the agents are capable of performing multiple tasks. A UAV can have a hybrid manipulator capable of performing ball-grabbing and balloon-popping tasks and thus can participate in other tasks after finishing its first assigned task. Situations may arise where agents are redundant. One such example is when two UAVs are deployed for grabbing the ball, and one of them finishes its task. In such cases, OMS identifies the redundant task and associated UAV. Further, it reassigns that particular agent to another task, speeding up the latter.
3. **State monitoring and fail-safe mechanism:** The master receives the information from all its agents, thus helping it monitor their states (position, velocity, etc.) apart from the status of their tasks. This ensures that a smooth reallocation of tasks can be achieved in the event of agent failure or redundancy. In case of any unexpected situations, a fail-safe mechanism can help redirect the UAVs' control to a pre-programmed set of instructions. This will prevent crashes and safety breaches.

7. ROS Simulations

Owing to the complexity of the tasks, the inconvenience of field testing of three UAVs at once frequently, and rapid algorithm testing, we developed a virtual environment in Gazebo that simulates



(a) (b) (c) (d)

Figure 20. (a)-(c) Arena for Challenge 1 in Gazebo. The balloons are erected as per the specifications. The net enclosure of the competition arena is recreated using a custom model. The orientation of the arena is made similar to the venue (d) Iris drone with a sideways manipulator arm and camera attached to it.

the arena of Challenge 1 in terms of dimensions and objectives. The following aspects were simulated.

1. Base Arena: The arena of 100 m × 60 m × 20 m is created using a simple rectangle of the same dimensions (Figure 20a-20b). To simulate the nets, a cloth object around the arena was simulated physically in the 3D modeling software Blender and then exported to Gazebo. The whole arena is tilted by 18 degrees with respect to magnetic North, to match with the inclination of the Challenge 1 arena at ADNEC, Abu Dhabi (Figure 20c).
2. Balloons: Six balloons of prescribed size and color are erected, and the wind effect is emulated using a sinusoidal varying force applied to the balloon link, in random directions. This gives the desired sway to the balloons. The sway is facilitated by the presence of a ball joint between the balloon link and the rectangular base.
3. Target UAV: The target UAV used is 3DR Iris. The ball of 0.1 m diameter is hung from this UAV using a thin cylindrical link of the specified dimensions by a ball joint so that the ball sways freely as it would on the real UAV. The target UAV repeatedly moves in a figure-of-eight path, whose major and minor axes, yaw, pitch, and roll, can be configured.
4. Interceptor UAVs: The interceptor UAVs are again 3DR Iris, with the manipulation mechanism fixed at the side using a fixed joint. To simulate the grabbing manipulator, a flat plate of approximate size to the real grabber is attached at 0.6 m from the UAV center of gravity (Figure 20d). It also has a simulated camera on the flat plate, as in the real manipulator. The simulated camera has the same field of view and resolution as the real one.

The ROS node communication architecture for the three UAVs, is shown in Figure 21. In the figure, oval shapes denote the ROS nodes, while the rectangular shapes denote the ROS topics being published and subscribed from these nodes. They are further grouped according to their namespaces. The topics belonging to each UAV are assigned the corresponding color to visualize the data flow.

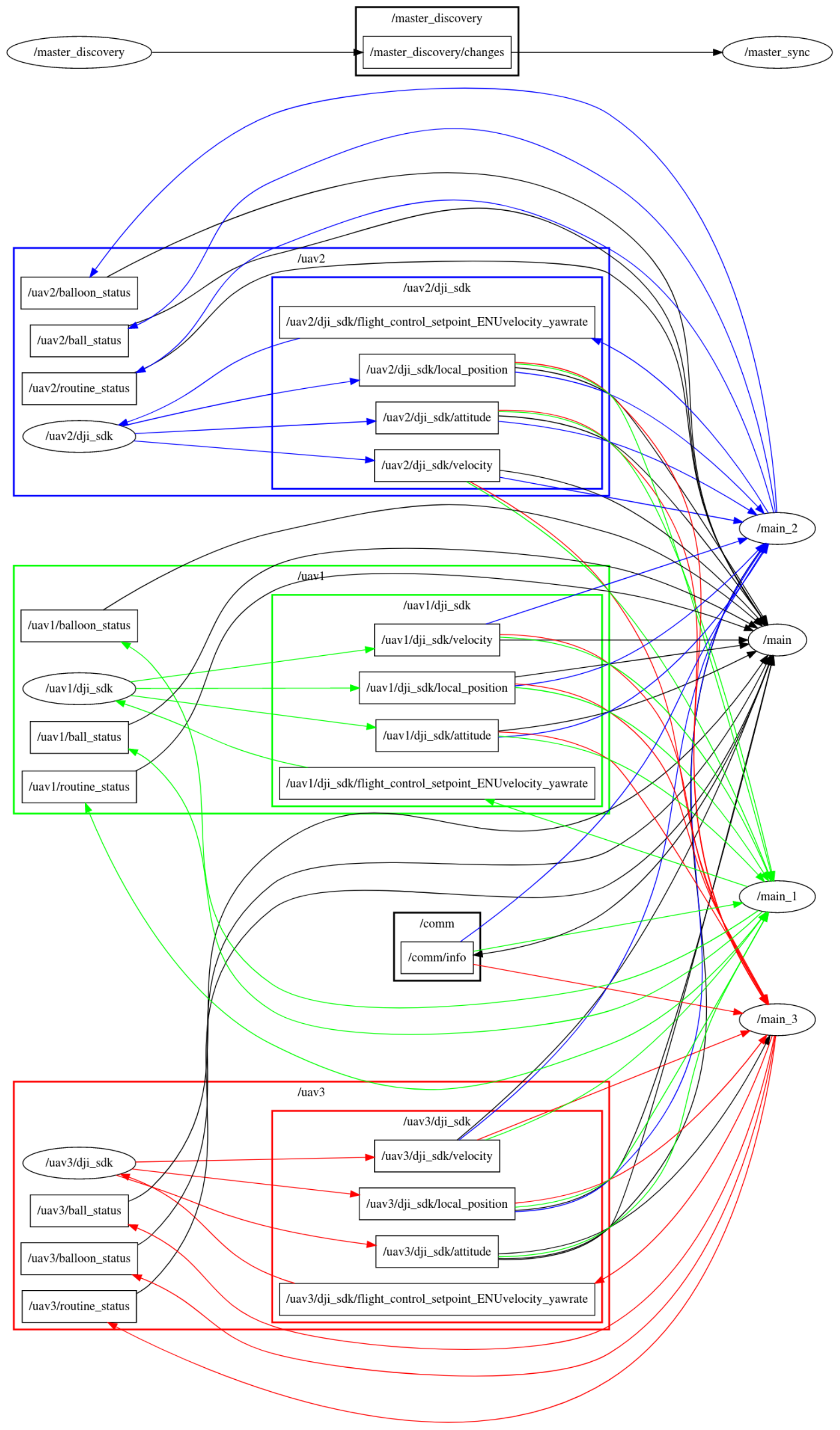


Figure 21. Network representation of the ROS nodes involved in the operation manager.

It can be seen that each UAV (main_1, main_2, and main_3) is listening to the position, velocity, and orientation topics of the other UAVs for collision avoidance calculations. The Ground Control Station node (main) listens to all of these plus the individual task state topics reported by each local node (ball_status and balloon_status) and issues tasks on the routine_status topic.

Simulations are performed to execute the complete mission and task allocation as per the mission described in Figure 22. Two UAVs cooperatively grab the ball while a single UAV achieves the balloon-popping task. The flow of control among the UAVs are shown in the flowchart. The trajectory of three UAVs is plotted in Figure 23, where the different color indicates the different tasks for the

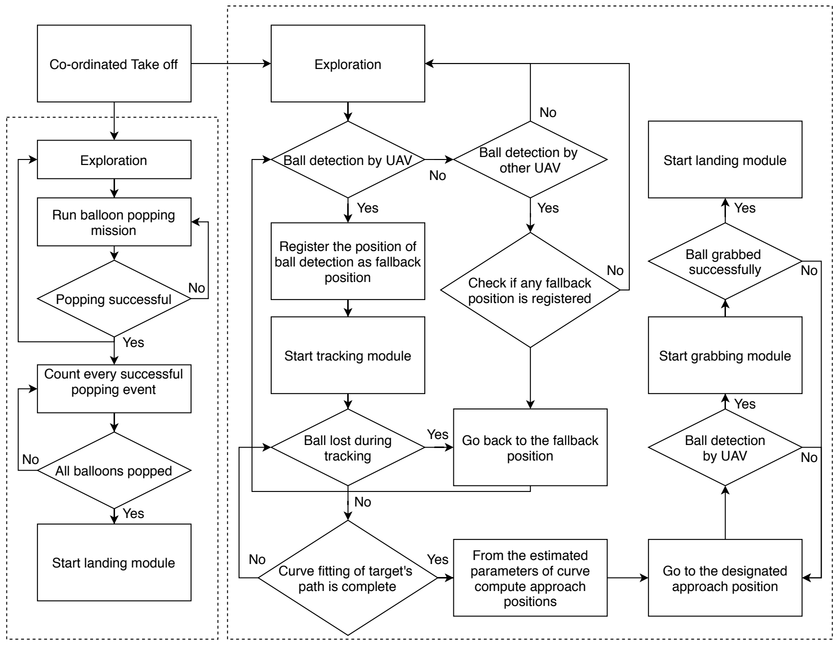

Co-ordinated Take off
Exploration
Run balloon popping mission
Popping successful
No
Yes
Count every successful popping event
All balloons popped
No
Yes
Start landing module
Exploration
Ball detection by UAV
No
Yes
Ball detection by other UAV
No
Yes
Register the position of ball detection as fallback position
Check if any fallback position is registered
No
Start tracking module
Ball lost during tracking
Yes
Go back to the fallback position
No
Curve fitting of target's path is complete
No
Yes
From the estimated parameters of curve compute approach positions
Go to the designated approach position
Ball detection by UAV
Yes
No
Start grabbing module
Ball grabbed successfully
Yes
No
Start landing module

Figure 22. Flow of control for the complete mission.

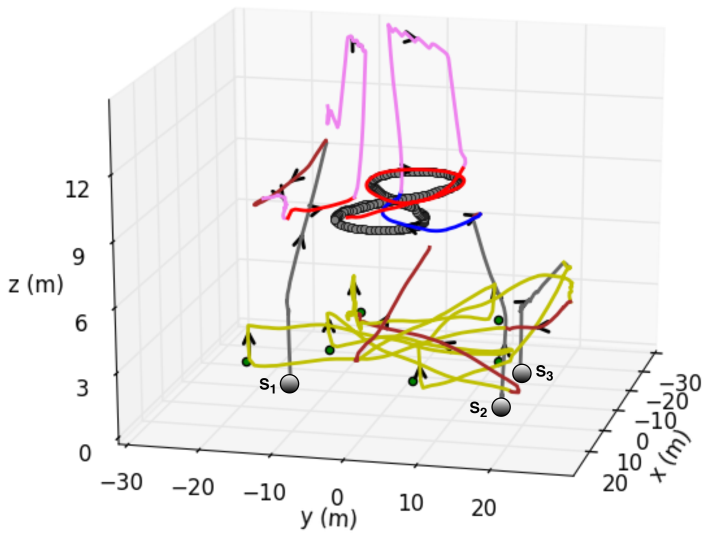


Figure 23. Task allocation during complete mission. The color codes correspond to the task switching as shown in Figure 24.

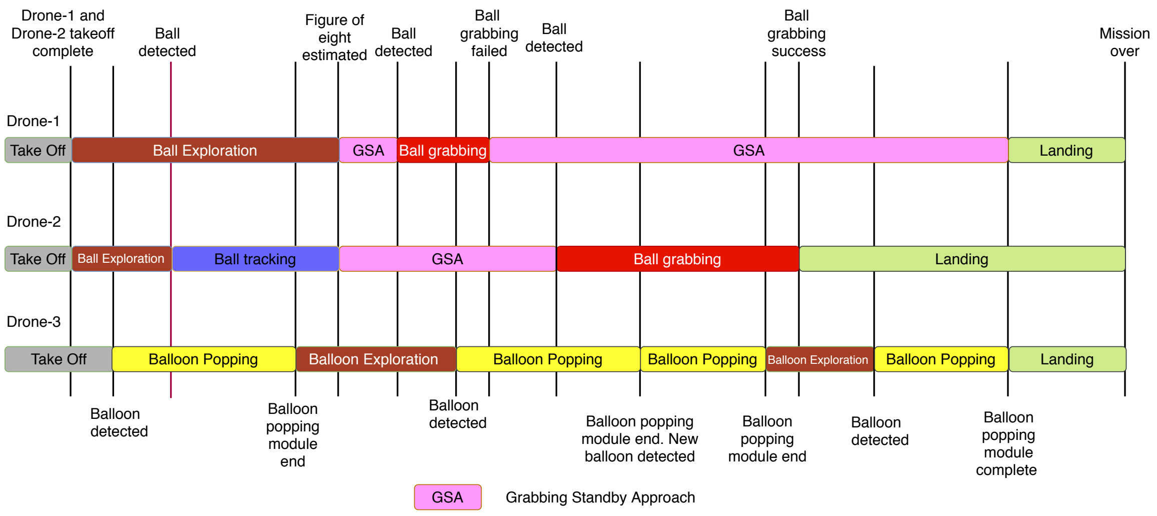


Figure 24. Task switching showing the switching of control between different tasks in the OMS.

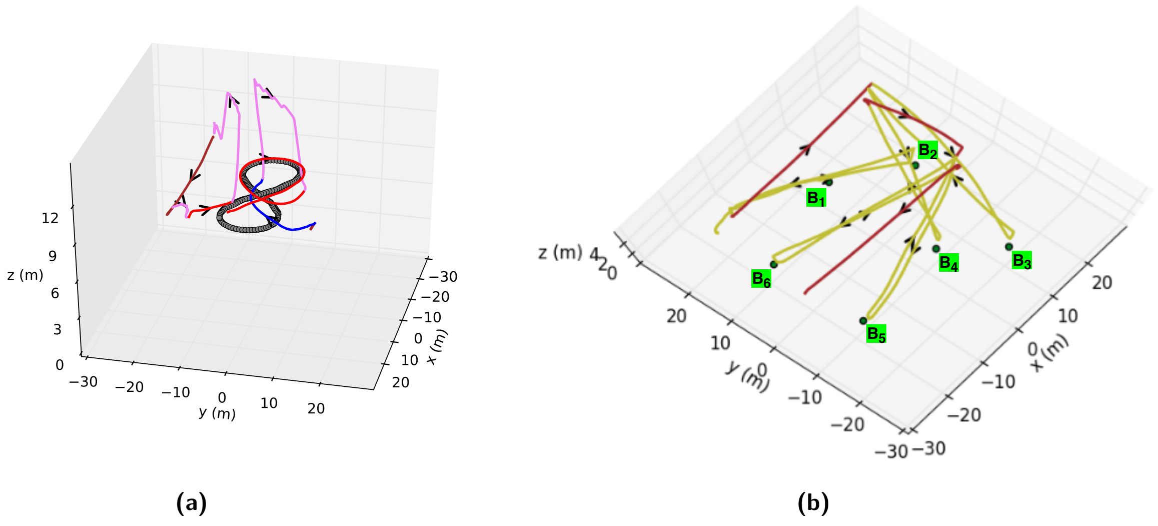


Figure 25. (a) Ball-grabbing task (b) Top view of balloon-popping task.

UAVs as commanded by OMS. S1, S2, and S3 are the take-off points for UAV 1, UAV 2, and UAV 3, respectively. The allotted tasks by OMS among the different agents are presented through the task switching diagram, shown in Figure 24, and corresponding trajectories (Figure 23) of the UAVs are plotted in the same color. UAV 1 and UAV 2 are initially allocated static tasks of ball exploration after take-off, while UAV 3 is allocated the static task of balloon exploration. The vertical lines in Figure 24 indicate the switching of a task due to the completion of a task or a special event like a ball detection or failure to grab. The trajectories of the UAVs specifically during ball-grabbing and balloon-popping are plotted in Figure 25a and Figure 25b. In Figure 25b, B_1-B_6 are the different balloon locations.The balloons are popped in the order from B_1 to B_6. The simulation videos[1] demonstrate the mission execution and performance of the developed system in the Gazebo environment.

[1] Simulation videos

Table 2. Design specifications

Specifications	Requirements
Max size	1.2 m × 1.2 m × 0.5 m
Endurance	At least 15 minutes
Manipulator location	Sideways
Gripper	Fixed to manipulator
Payload requirement	At least 1-2 kg

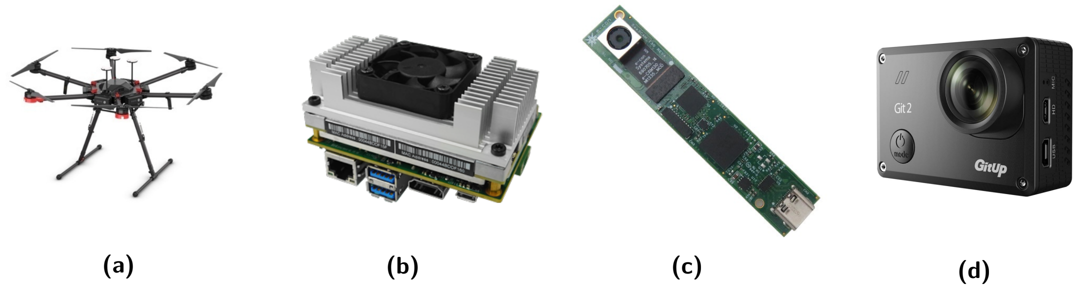


Figure 26. (a) DJI M600 hexacopter (b) NVIDIA Jetson TX2 with Auvidea J130 carrier board (c) See3 130 camera (d) Git2P action camera.

8. Hardware Design

As per the specifications and design considerations mentioned in Section 1 and Section 3, the basic requirements for the interceptor UAV along with manipulator is listed in Table 2. Apart from these, the ball-grabbing UAV has to be stable during the grabbing process, as it needs to exert a force of 4 N for the detachment of the ball. Also, the ball-grabbing UAV has to withstand the downwash from the target UAV during the grabbing process. To satisfy the above specifications, the DJI M600 Pro is selected. It is a hexacopter with triple-redundant GPS and 18 minutes of flight time for 5 kg payload. The usage of the hexacopter aids better stability while grabbing the ball, where quick maneuvers are desired. The different equipment used for Challenge 1 are as below.

1. **UAV and on-board computers:** As mentioned above, DJI M600 Pro (Figure 26a) is used as the UAV for grabbing and popping. The UAV is commanded via the DJI A3 Pro flight controller, which is a highly reliable autopilot. The companion computer is NVIDIA Jetson TX2 (Figure 26b). This computer is mounted on an Auvidea J130-2k4k carrier board. This is a well-equipped carrier board with four USB ports, 1 GB/s ethernet port, and HDMI input and output. This helps accommodate all the sensors on-board with ease. An Arduino Mega 2560 is used to control the manipulator's arm and an Arduino Nano for the propeller guards.
2. **Vision sensors:** The target detection is achieved using visual feedback. The vision sensor employed for target tracking and interception is a See3CAM 130 (Figure 26c), an industrial-grade 4K USB monocular camera. This camera's choice helped in better detection of the ball from long distances and for a variety of backgrounds. This camera has an added advantage of smaller weight and size, which is essential for it to be mounted on the manipulator. The exploration was carried out using Gitup Git2 Pro (Figure 26d). This is an action camera with 170° field of view, which aids in better visibility of the target ball in the arena. A T-3D V metal 3-Axis brushless gimbal is attached to the front of the UAV to hold the Gitup camera in place.
3. **Manipulation mechanism and communication system:** An entirely in-house designed and fabricated one degree of freedom manipulator is employed for the challenge. Increasing the degrees of freedom provides a larger operational space for the end effector. However, this, in turn, brings in delay in actuation, more complexity in control of the manipulator, and

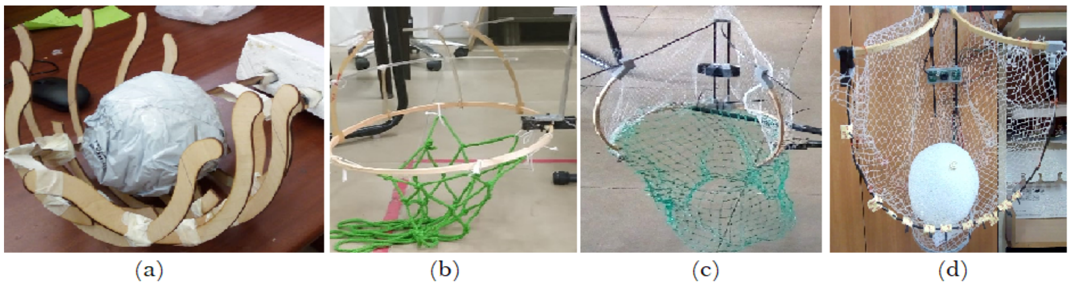


Figure 27. Evolution in design of the basket type manipulator (a) Active gripper (closes to detach) (b) Passive end-effector (the crown detaches ball at any of the multiple detachment points) (c) Passive semi-open basket with dedicated camera mount (single detachment point at the center) (d) Multi-utility (grab/pop) passive semi-open basket with eye-in-hand configuration.

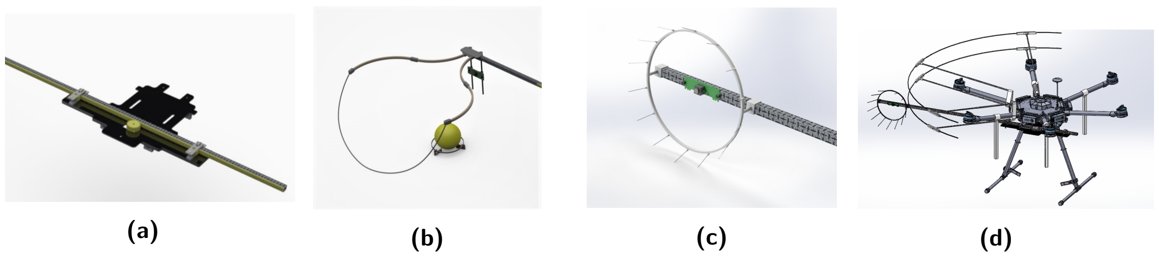


Figure 28. The relevant design components of the manipulation mechanism are (a) The rack and pinion mechanism (linear actuation) with its mount (b) The end effector to grab ball (c) The end effector for popping balloons (d) The manipulator along with propeller guards.

increases power consumption. Therefore, a single degree of freedom manipulation mechanism was developed for the Challenge 1 tasks.

The design mechanism evolved as shown in Figure 27. The initial idea was that of an active manipulator (Figure 27a), a twin claw system to detach the ball, made of balsa wood. The upper portion of the gripper closes to detach the ball, and the lower portion opens to drop the ball. To make the mechanism energy optimal and lightweight, a passive basket was conceived of next (Figure 27b), which retained the finger like multiple detachment mechanism from the previous version in the form of acrylic but was remodeled and fixed to a wooden basket, like its crown. But the optimal placement of the camera and a clear field of view was difficult to obtain with this configuration. Besides, the front rim lowered the chances of successful grabbing. This led to the development of a newer design (Figure 27c), which adopted a single point of detachment mechanism rather than the many points of detachment from the previous design. This was also a semi-open basket with dedicated mounts for the camera (eye-in-hand). After extensive testing, the final design was arrived at (Figure 27d), where the gripper was modified such that it could grab the ball as well as pop the balloon. Sharp needles were attached to the lower portion of the basket to aid popping while the top portion of the gripper executed the ball detachment. The detachment portion is made of lightweight wood, while the lower portion of the gripper is made of thin carbon fiber strips. The basket net for collecting the ball for the final design is made from a fishing line. This made the setup lightweight and also offered less drag.

The manipulator used for the competition was fabricated using carbon fiber, which provides better strength to weight ratio. To be within the size constraint, the manipulator is kept retracted under the UAV and actuated linearly. The in-flight extension of the arm is achieved using a rack and pinion mechanism (Figure 28a). The range of extension is calculated to ensure stability and zero moments on the overall system while executing maneuvers. The end effector of this manipulator is a passive basket with a fruit-picker-like mechanism to detach the ball (Figure 28b). A lightweight net is attached to the rim of this basket to collect the ball. Limit switches are attached to the bottom of this passive basket, as seen in (Figure 28b). This acts as the feedback mechanism for detecting successful grabbing of the ball. The camera is fixed to the center of the basket, which results in an eye-in-hand configuration. This is especially useful in the approach and grab phases of the challenge.

For popping balloons, a dedicated mechanism is employed. The manipulator's arm works similar to the ball grabber while the end effector has a structure shown in (Figure 28c). Sharp needles are arranged on a mount, which is 3D printed using PLA. Here too, an eye-in-hand configuration is used to help pop the balloon.

Propeller guards (Figure 28d) are incorporated to ensure safety while carrying out the tasks. The propeller guard is designed to extend in-flight. Suitable 3D printed mounts are attached to the UAV arms to support the extension of the guards after take-off. The manipulator actuation and propeller guard extensions are carried out using an on-board Arduino Mega board. A detailed account of manipulator design and development is given in (Vidyadhara et al., 2021b; Vidyadhara et al., 2021a).

5GHz Wi-Fi set up is installed for field experiments, as it has better bandwidth than the 2.4 GHz counterpart. Jetson TX2 can connect to 2.4 GHz and a lower band of 5 GHz Wi-Fi. For testing purposes, TL-MR3020, a portable 3G/4G DC wireless network router, set to a static channel is used. The ground control station is an Alienware m15 R1 laptop. Its GPU is utilized for training the vision algorithms as well.

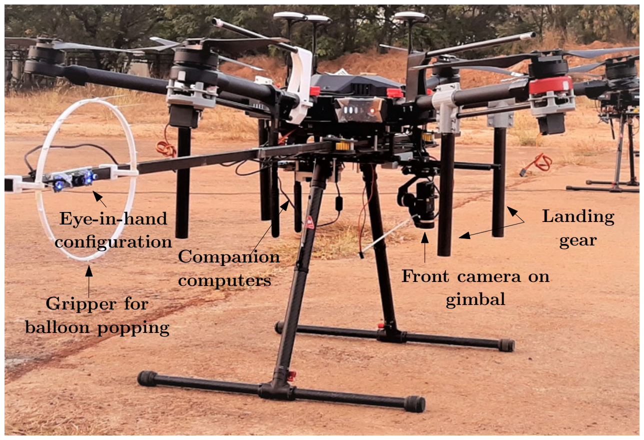
9. Experimental Results

The following section details the flight experiments conducted at the IISc airfield and those during the competition. The results achieved and the major observations are reported in this section. As per the earlier competition rules, the color of the target ball was red, and the balloon was white. Therefore, some of the field experiences are performed with a red ball and white balloons.

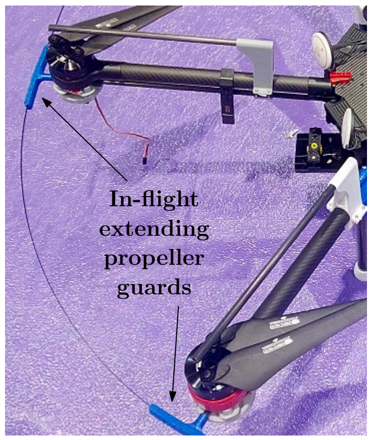
9.1. Field experiments

Simulations were performed using the PX4 autopilot in the Gazebo environment. The DJI based system was used for the competition and ROS-Gazebo was used to simulate the OMS and further deploy it on hardware. By doing so, the OMS developed on the simulator can be seamlessly transferred to the DJI environment with little to no modifications. The final configuration of the balloon-popping UAV and ball-grabbing UAV are shown in Figure 29a and Figure 29c. The interceptor UAVs are equipped with propeller guards (Figure 29b).

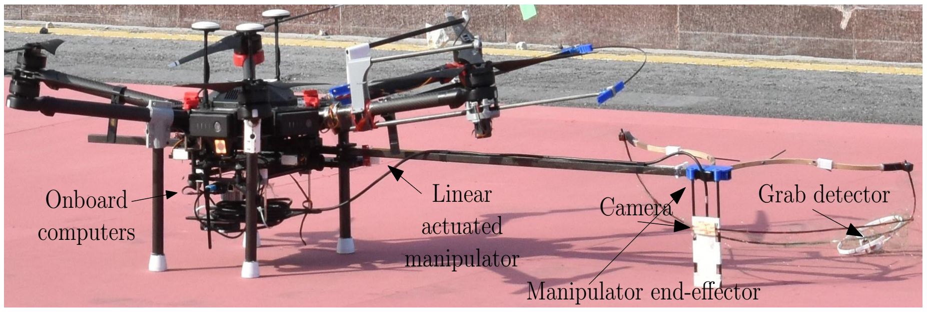
Individual algorithms are initially tested and the various parameters such as camera parameters, tracking distance, adaptive controller gains, etc., are tuned for the outdoor testing. The flight results for a successful ball-grabbing scenario is described here. The variation of different system states during the ball-grabbing experiment, for an initial target depth of approximately 10 m, is shown in Figure 30. The variation of target pixel coordinates in the camera frame during the engagement is shown in Figure 30a. The variation in the ball depth during the outdoor grabbing engagement is shown in Figure 30b. Clearly, the ball depth goes to zero for successful grabbing. As the ball depth is measured from the visual information, the ball depth does not reduce smoothly as observed in simulation results. The desired yaw rate obtained from the guidance strategy is plotted in Figure 30c, where the different colors correspond to the different pixel trajectories given in Figure 30a. The tracking of commanded velocities and yaw rate by the interceptor UAV is shown in Figure 30d-30g. As it is evident from the plots, the system is able to track the commanded values. The variation of the attitudes during the grabbing engagement is shown in Figure 30h-30j. The corresponding attitude rates are shown in Figure 30k-30m. As can be observed from these, the vehicle remains



(a)



(b)



(c)

Figure 29. (a) UAV set-up for balloon-popping (b) The in-flight extendable propeller guard (c) The grabber UAV with passive basket type single DoF manipulator.

stable during the entire mission. Figure 31 and Figure 32 show the relevant frames of grabbing task and balloon-popping task, respectively.

After the validation of individual algorithms, the overall system architecture is verified through SITL and HITL. The successful testing is followed by deployment of the hardware described in Section 8. One UAV (Figure 29a) is attached with the balloon-popping mechanism, while two UAVs are attached with the basket-type passive end-effector (Figure 29c). The operation manager is initialized with the launch and mission parameters. An end-to-end test on the flight modes was conducted, the results of which are included here. Numerous flight tests were carried out to evaluate the performance of the overall system. The task achieved success 7 out of 10 times. The unsuccessful cases were those which operated in high wind conditions or faced failure in detection which was rectified later. Additional experimental data for two more successful tasks are shown in Figure 33 and Figure 34. Each of these are for different initial conditions of the UAVs. The corresponding flight test videos[2] demonstrate the real time performance of the developed framework.

9.2. Competition experiences

The competition arena was small in size compared to the originally announced dimensions. The speed of the target UAV was specified to be low/high without any quantitative mentions. The standby area of our architecture, for the interception, was adjusted based on the location of the

[2] Experiment videos

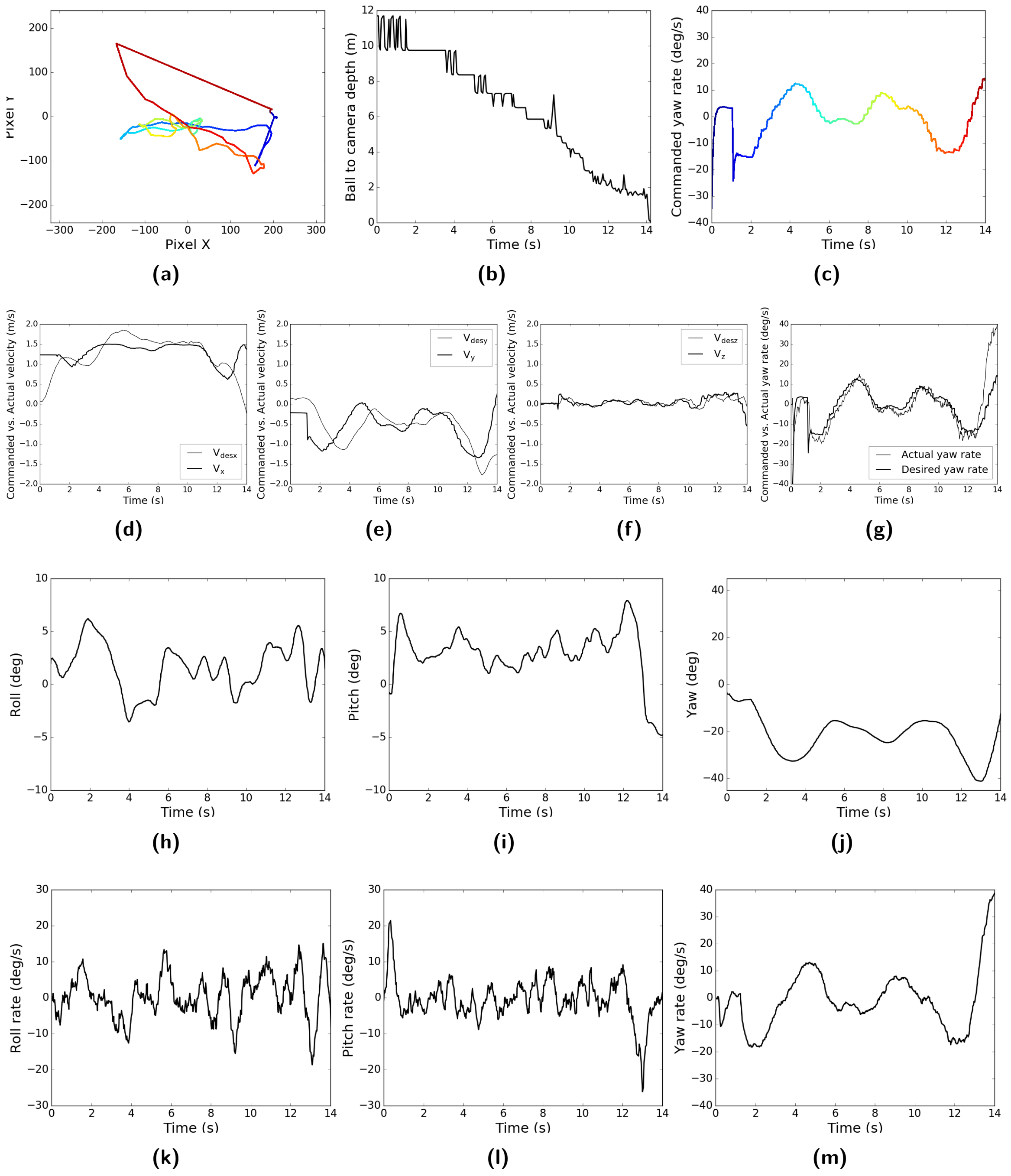


Figure 30. Experiment 1 (a) Variation of target pixel in camera frame (b) Variation of ball depth from camera during grabbing (c) Commanded yaw rate during grabbing phase. Commanded vs. actual velocity in (d) *x* direction (e) *y* direction (f) *z* direction (g) Commanded vs. actual yaw rate (h) Roll angle (i) Pitch angle (j) Yaw angle of the interceptor UAV while in grabbing task (k) Roll rate (l) Pitch rate (m) Yaw rate of the grabber UAV. The rates confirm stable engagement.

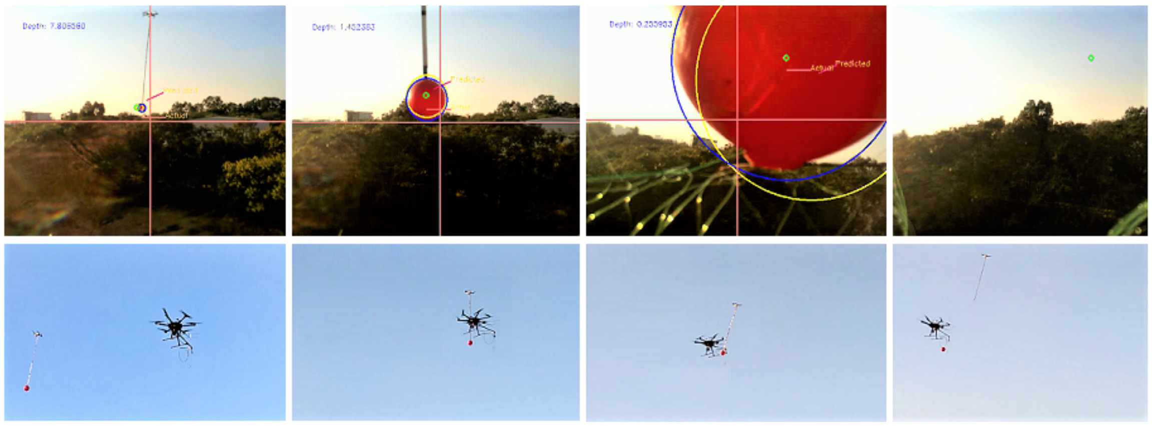


Figure 31. The ball-grabbing experiment, visualised frame by frame, from the UAV camera (top) and a third person (bottom). From left: ball being detected, approached, intercepted and detached. The initial experiments were carried out using a red ball.

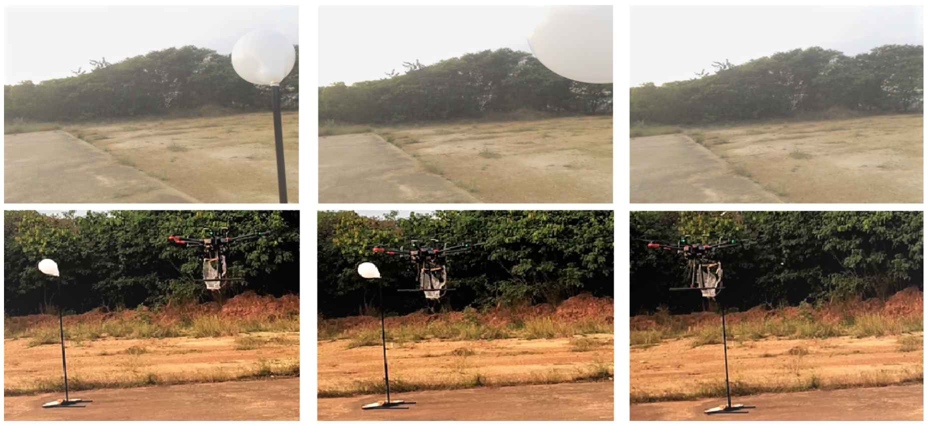


Figure 32. Balloon-popping experiment from the UAV camera (top) and third person (bottom). From left: detection, approaching and popping. The experiments were carried out using a white balloon.

sun during the time of competition, for improved performance of the vision module. The weather was very sunny during the day. The initial position of the UAVs and the angle of interception for the task were set based on the time of the day to ensure that the cameras were facing away from the sun to minimize the glare and visual artifacts that could be caused due to lens flare. Several task planning parameters like the level of exploration were adjusted, as full 25 m height was not available in the arena due to the sag of enclosing net. Also, the task parameters were made more conservative due to the huge uncertainty in the GPS location accuracy at the venue. This was due to a large number of steel structures around the arena. The communication band of our system was not fully compatible with the network provided by the organizers during the competition. So, it was not possible to establish multi-UAV communication architecture properly in the arena. The code structure was modified to perform individual tasks without inter-agent communication; however, it was not robust and efficient like the previously developed architecture. We could only perform some parts of the mission. Few snapshots of the task during MBZIRC 2020, are shown in Figure 35 and relevant video can be found in the link[3] below.

[3] MBZIRC20 videos

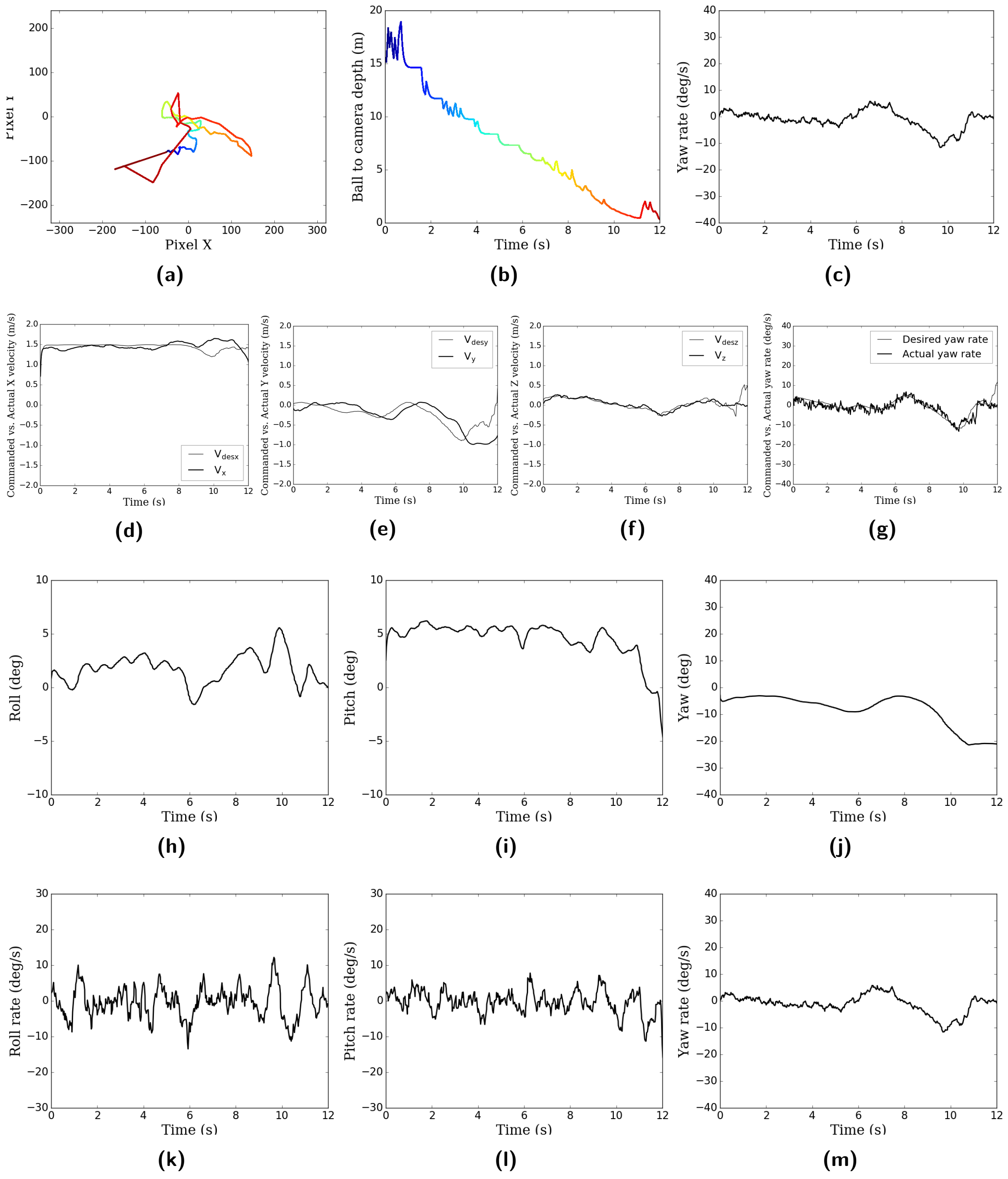


Figure 33. Experiment 2 (a) Variation of target pixel in camera frame (b) Variation of ball depth from camera during grabbing (c) Commanded yaw rate during grabbing phase. Commanded vs. actual velocity in (d) x direction (e) y direction (f) z direction (g) Commanded vs. actual yaw rate (h) Roll angle (i) Pitch angle (j) Yaw angle of the interceptor UAV while in grabbing task (k) Roll rate (l) Pitch rate (m) Yaw rate of the grabber UAV.

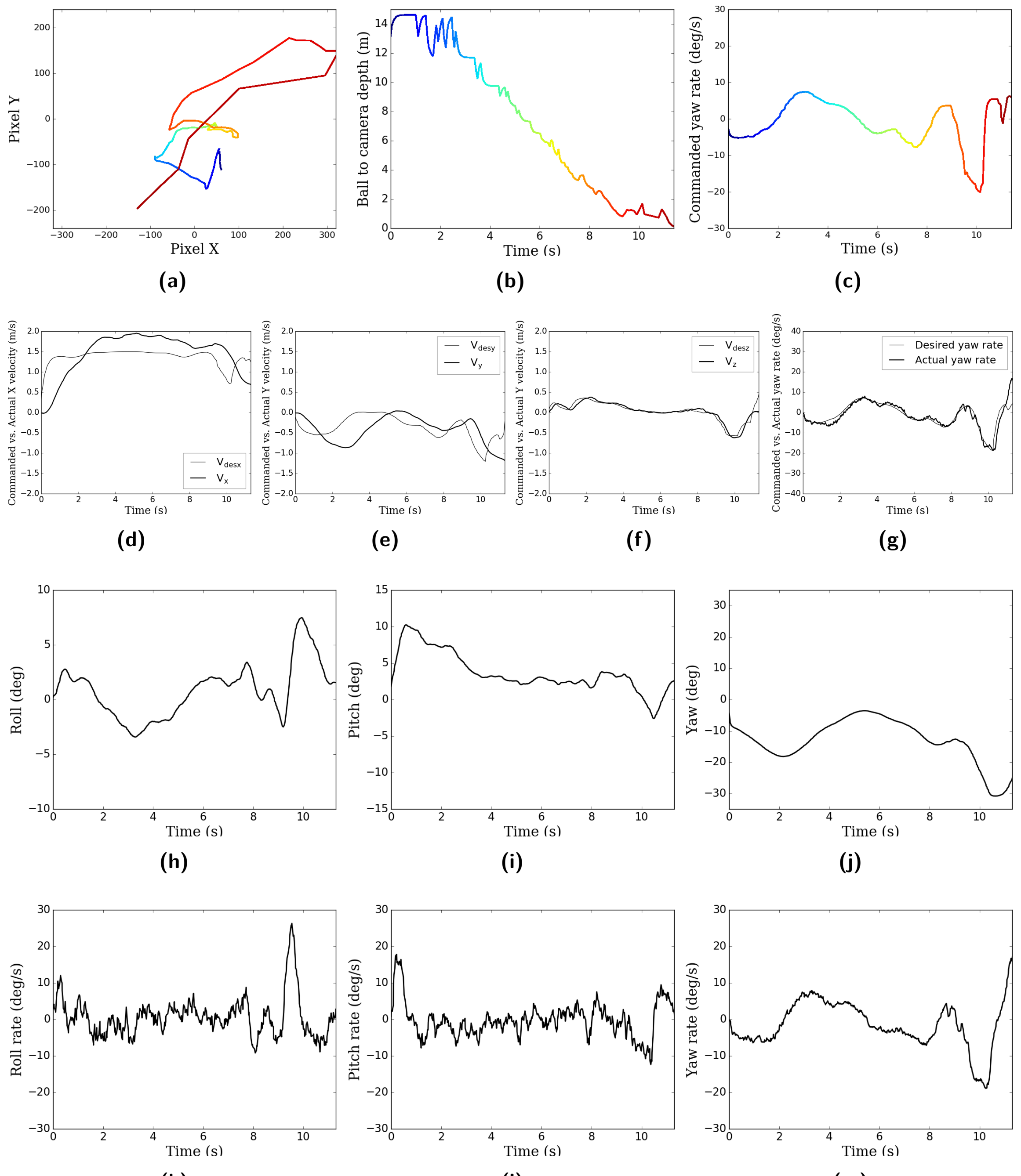
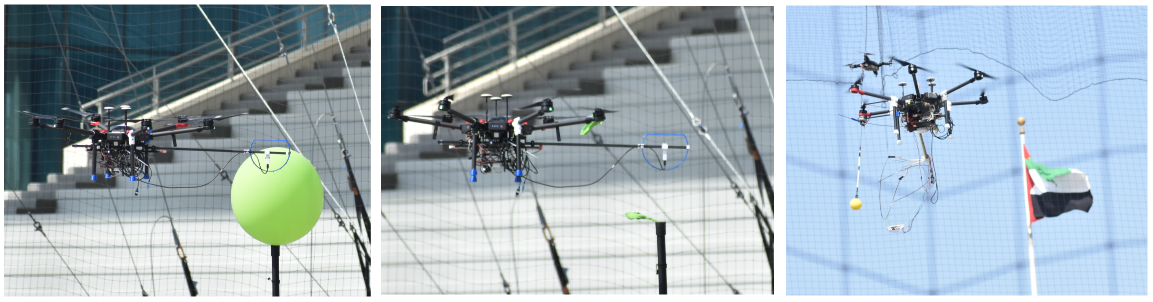


Figure 34. Experiment 3 (a) Variation of target pixel in camera frame (b) Variation of ball depth from camera during grabbing (c) Commanded yaw rate during grabbing phase. Commanded vs. actual velocity in (d) x direction (e) y direction (f) z direction (g) Commanded vs. actual yaw rate (h) Roll angle (i) Pitch angle (j) Yaw angle of the interceptor UAV while in grabbing task (k) Roll rate (l) Pitch rate (m) Yaw rate of the grabber UAV.



(a) (b) (c)

Figure 35. From MBZIRC 2020 (a) UAV approaching to pop the balloon (b) Successful popping (c) UAV approaching to grab ball.

10. Discussions

The main tasks of the autonomous mission involving aerial interception and grabbing of stationary and moving targets using visual information in an unknown outdoor environment are addressed in this work. Although aerial interception using visual information is highly affected due to environmental conditions, accuracy is improved with an efficient manipulator system integrated with robust guidance strategy. The developed system is able to grab the ball with a high success rate from the target moving at a speed of 5 m/s. We found the grabbing success rate in tail-chase mode higher than the head-on mode, since the engagement duration is comparatively shorter in the latter. The system was able to track a target moving in a figure-of-eight trajectory at a speed of 4 m/s at a distance of 8 m. At higher tracking distance, the depth from the camera is not sufficiently robust for consistent tracking. It was found difficult to track the target in the curved portions of the figure-of-eight trajectory when the target speed exceeded 4 m/s.

The following critical lessons were learned during system development and outdoor flight test experiments.

1. In the external environment, the delay and the uncertainty in the visual sensor data can play a significant role in a UAV task involving moving objects. The overall system design should accommodate those factors through hardware design or algorithm design. The aerial manipulator should be designed iteratively after getting feedback from the flight test results and visual-guidance system.
2. In outdoor conditions, vision algorithms involving target depth should be avoided for improved system performance in grabbing dynamic targets.
3. The centralized command center for task allocation during a multi-UAV task will have better efficiency compared to decentralized architecture; however, the whole mission can fail in the case of inter-agent communication breakdown. The software architecture should encompass fail-safe scenarios, so that individual agents can complete their tasks without necessarily halting the entire mission.
4. The solution for the vision tasks, when designed solely using deep neural networks, can generalize well and address the corner cases. End-to-end, deep neural network designs can detect and track objects by learning complex and robust features, when trained properly.
5. GPUs with specialized, deep learning capabilities, such as Jetson Xavier processors, can significantly boost the deep-learning models in terms of inference speed.
6. Augmenting the training data (using transformed or synthetically-generated data) while training the deep-learning model can facilitate the use of a different set of training samples in each epoch. This way, when trained adequately, the model performs better than a model trained with the same (unaugmented) dataset in all epochs.

11. Conclusions

This paper presented the software and hardware developed to address the problems in Challenge 1 of MBZIRC 2020. The challenge required to autonomously detect, track, and grab a ball from a maneuvering target as well as autonomously detect and approach stationary balloons and burst them. A custom-designed and developed manipulation mechanism are employed to achieve the tasks of grabbing and popping. The sideways design of the manipulator ensured safe detachment of the ball and popping of balloons. The passive end-effector design also reduced the overall system energy requirements. A vision-based guidance strategy was formulated to ensure the ball-grabbing. Two UAVs were used to coordinate between themselves to explore, track, and grab the ball. Popping was executed using a similar guidance philosophy, with minor modifications to suit the purpose. Underlying relations governing the algorithms were also presented in detail. The entire mission was handled by an Operation Management System (OMS), which executed multi-UAV task allocation and switching. The OMS architecture also included various features such as state monitoring, fail-safe definitions, and task reallocation. The framework was developed in ROS. Simulation results of the developed solution were presented to demonstrate the performance of the developed solution in the virtual environments. The results of field experiments were also included, demonstrating the real time implementation of the framework and its efficiency in dynamic environment. The techniques developed in addressing the challenge have far more generality in applications that go beyond the specified tasks. They can be used in many critical applications indoors and outdoors, where autonomous systems are needed to carry out aerial manipulation tasks. The developed integrated system can be employed as counter-UAV technology in warfare. The manipulators with minor modification could be used for applications like package passing between drones in long distance delivery, repair and monitoring of inaccessible structures, fruit picking in orchards, among many others.

12. Index to Multimedia Extensions

The following videos are available as supporting information in the online version of this article.

Extension	Media type	Description
1	Video	ROS-Gazebo simulations
2	Video	Field-experiments
3	Video	Performance at competition

Author credits

Lima Agnel Tony: Integration of different systems, project management, sub-team coordination and interface, concept development, pilot, writing the paper.

Shuvrangshu Jana: Concept development, guidance and estimation codes, mission planner, software architecture, writing the paper.

Varun V.P.: Coding support, Gazebo SITL simulation, GCS management, writing the paper.

Aashay Anil Bhise: Guidance, control, and estimation software, geo-fencing codes, mission planner development, code stack finalization, Gazebo SITL simulation, GCS management, writing the paper.

Aruul Mozhi Varman S.: Vision code development, writing the paper.

Vidyadhara B. V.: Manipulators and propeller guard design and prototyping, design of component mounts and landing gears.

Mohitvishnu S. Gadde: Main pilot, electronics integration, communication set-up, mission planner.

Raghu Krishnapuram: Lead in vision algorithm design and implementation, writing the paper.

Debasish Ghose: Overall project lead, conceptualization, writing the paper.

Acknowledgements

We would like to acknowledge the Robert Bosch Center for Cyber Physical Systems, Indian Institute of Science, Bangalore, and Khalifa University, Abu Dhabi, for partial financial support. We would also like to thank fellow team members from IISc and all members of Guidance, Control, and Decision Systems Laboratory, especially, Abhishek Kashyap, Rahul Ravichandran, Aakriti Agrawal, Rohitkumar Arasanipalai, Kuntal Ghosh, Amit Singh, Kumar Ankit and Narayani Vedam for their valuable contributions towards this competition. We also thank Mayur Shewale and Vishal Pal for their valuable time. We appreciate all the help and support from our collaborator, Tata Consultancy Services.

ORCID

Lima Agnel Tony[1] https://orcid.org/0000-0002-5173-9089
Shuvrangshu Jana[1] https://orcid.org/0000-0001-8906-8519
Varun V. P.[2] https://orcid.org/0000-0002-7772-6833
Aashay Anil Bhise[1] https://orcid.org/0000-0003-0716-3083
Aruul Mozhi Varman S.[2] https://orcid.org/0000-0003-4414-6514
Vidyadhara B. V.[1] https://orcid.org/0000-0002-1037-8010
Mohitvishnu S. Gadde[1] https://orcid.org/0000-0002-3566-4275
Raghu Krishnapuram[2] https://orcid.org/0000-0003-2979-8186
Debasish Ghose[1] https://orcid.org/0000-0001-5022-4123

References

Agrawal, A., Arasanipalai, R., and Ghose, D. (February, 2020). Least squares curve fitting and RNN-based trajectory estimation for noisy looping trajectories. In *Proceedings of Mohamed Bin Zayed International Robotics Challenge Symposium (MBZIRCS-2020)*. Khalifa University.

Alexis, K., Darivianakis, G., Burri, M., and Siegwart, R. (2016). Aerial robotic contact-based inspection: planning and control. *Autonomous Robots*, 40(4):631–655.

Antonelli, G., Baizid, K., Caccavale, F., Giglio, G., Muscio, G., and Pierri, F. (2014). Control software architecture for cooperative multiple unmanned aerial vehicle-manipulator systems.

Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483.

Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468.

Bhise, A. A., Jana, S., Tony, L. A., and Ghose, D. (February, 2020). Target state estimation and prediction for high speed interception. In *Proceedings of Mohamed Bin Zayed International Robotics Challenge Symposium (MBZIRCS-2020)*. Khalifa University.

Bolme, D. S., Beveridge, J. R., Draper, B. A., and Lui, Y. M. (2010). Visual object tracking using adaptive correlation filters. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2544–2550.

Bouffard, P., Aswani, A., and Tomlin, C. (2012). Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results. In *2012 IEEE International Conference on Robotics and Automation*, pages 279–284. IEEE.

Chakravarthy, A. and Ghose, D. (2012). Generalization of the collision cone approach for motion safety in 3-d environments. *Autonomous Robots*, 32(3):243–266.

Chaumette, F. and Hutchinson, S. (2006). Visual servo control. i. basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90.

Cheung, Y., Huang, Y.-T., and Lien, J.-J. J. (2015). Visual guided adaptive robotic interceptions with occluded target motion estimations. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6067–6072. IEEE.

Clark, M., Kern, Z., and Prazenica, R. J. (2015). A vision-based proportional navigation guidance law for uas sense and avoid. In *AIAA guidance, navigation, and control conference*, page 0074.

Danelljan, M., Bhat, G., Khan, F. S., and Felsberg, M. (2017a). Eco: Efficient convolution operators for tracking. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6931–6939.

Danelljan, M., Häger, G., Khan, F. S., and Felsberg, M. (2015). Learning spatially regularized correlation filters for visual tracking. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4310–4318.

Danelljan, M., Häger, G., Khan, F. S., and Felsberg, M. (2017b). Discriminative scale space tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8):1561–1575.

Dong, G. and Zhu, Z. H. (2016). Autonomous robotic capture of non-cooperative target by adaptive extended kalman filter based visual servo. *Acta Astronautica*, 122:209–218.

Dwibedi, D., Misra, I., and Hebert, M. (2017). Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *The IEEE International Conference on Computer Vision (ICCV)*.

Dwiyasa, F., Lim, M.-H., Kang, P., Foo, R.-X., and Teo, S.-W. J. (2020). Heterogeneous multi-robot mission planning for coordinated tasks execution. In *Soft Computing for Problem Solving 2019*, pages 167–173. Springer.

Efremov, M. A. and Kholod, I. I. (2020). Architecture of swarm robotics system software infrastructure. *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–4.

Floroian, D., Ursutiu, D., Floroian, L., and Moldoveanu, F. (2010). Robosmith: Wireless networked architecture for multiagent robotic system. *Int. J. Online Eng.*, 6:14–19.

Fumagalli, M., Naldi, R., Macchelli, A., Forte, F., Keemink, A. Q., Stramigioli, S., Carloni, R., and Marconi, L. (2014). Developing an aerial manipulator prototype: Physical interaction with the environment. *IEEE robotics & automation magazine*, 21(3):41–50.

Galoogahi, H. K., Sim, T., and Lucey, S. (2013). Multi-channel correlation filters. In *2013 IEEE International Conference on Computer Vision*, pages 3072–3079.

García, M., Viguria, A., Heredia, G., and Ollero, A. (2019). Minimal-time trajectories for interception of malicious drones in constrained environments. In *International Conference on Computer Vision Systems*, pages 734–743. Springer.

Gil, A., Aguilar, J., Dapena, E., and Rivas, R. (2019). A control architecture for robot swarms (ameb). *Cybernetics and Systems*, 50:300 – 322.

Gioioso, G., Ryll, M., Prattichizzo, D., Bülthoff, H. H., and Franchi, A. (2014). Turning a near-hovering controlled quadrotor into a 3d force effector. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6278–6284. IEEE.

Guerreiro, N. M., Butler, R. W., Maddalon, J. M., and Hagen, G. E. (2019). Mission planner algorithm for urban air mobility–initial performance characterization. In *AIAA Aviation 2019 Forum*, page 3626.

Jain, S., Xiong, B., and Grauman, K. (2017). Pixel objectness. *arXiv preprint arXiv:1701.05349*.

Jin, M., Yang, G., Liu, Y., Zhao, X., and Liu, H. (2018). A motion planning method based vision servo for free-flying space robot capturing a tumbling satellite. In *2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 883–888. IEEE.

Jones, E., Adra, D., and Miah, M. S. (2019). Mafoss: Multi-agent framework using open-source software. *2019 7th International Conference on Mechatronics Engineering (ICOM)*, pages 1–6.

Khalifa, U. (2020). *Mohamed Bin Zayed International Robotics Challenge-2020.*

Khamseh, H. B., Janabi-Sharifi, F., and Abdessameud, A. (2018). Aerial manipulation—a literature survey. *Robotics and Autonomous Systems*, 107:221–235.

Khan, G. H., Khan, S., Asad, M., Mehmood, Z., and Khan, U. (2018). A multi-target tracking and interception system in the surveillance area. In *2018 International Conference on Electrical Engineering (ICEE)*, pages 1–6. IEEE.

Kim, S., Seo, H., Choi, S., and Kim, H. J. (2016). Vision-guided aerial manipulation using a multirotor with a robotic arm. *IEEE/ASME Transactions On Mechatronics*, 21(4):1912–1923.

Kritsky, D., Ovsiannik, V., Pogudina, O., Shevel, V., and Druzhinin, E. (2019). Model for intercepting targets by the unmanned aerial vehicle. In *International scientific-practical conference*, pages 197–206. Springer.

Kumar, A., Ojha, A., and Padhy, P. K. (2017). Anticipated trajectory based proportional navigation guidance scheme for intercepting high maneuvering targets. *International Journal of Control, Automation and Systems*, 15(3):1351–1361.

Lee, D., Lim, H., Kim, H. J., Kim, Y., and Seong, K. J. (2012). Adaptive image-based visual servoing for an underactuated quadrotor system. *Journal of Guidance, Control, and Dynamics*, 35(4):1335–1353.

Li, M., Cai, Z., Yi, X., Wang, Z., Wang, Y., Zhang, Y., and Yang, X. (2016). Alliance-ros: A software architecture on ros for fault-tolerant cooperative multi-robot systems. In *PRICAI*.

Li, W., Zhao, R., Xiao, T., and Wang, X. (2014). Deepreid: Deep filter pairing neural network for person re-identification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 152–159.

Liu, C., Gong, S., Loy, C. C., and Lin, X. (2012). Person re-identification: What features are important? In Fusiello, A., Murino, V., and Cucchiara, R., editors, *Computer Vision – ECCV 2012. Workshops and Demonstrations*, pages 391–401, Berlin, Heidelberg. Springer Berlin Heidelberg.

Mebarki, R. and Lippiello, V. (2014). Image-based control for aerial manipulation. *Asian Journal of Control*, 16(3):646–656.

Mehta, S. S., Ton, C., Kan, Z., and Curtis, J. W. (2015). Vision-based navigation and guidance of a sensorless missile. *Journal of the Franklin Institute*, 352(12):5569–5598.

Nguyen, H.-N., Ha, C., and Lee, D. (2015). Mechanics, control and internal dynamics of quadrotor tool operation. *Automatica*, 61:289–301.

Papachristos, C., Alexis, K., and Tzes, A. (2014). Efficient force exertion for aerial robotic manipulation: Exploiting the thrust-vectoring authority of a tri-tiltrotor uav. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 4500–4505. IEEE.

Ramon-Soria, P., Arrue, B. C., and Ollero, A. (2020). Grasp planning and visual servoing for an outdoors aerial dual manipulator. *Engineering*, 6(1):77–88.

Ramon Soria, P., Bevec, R., Arrue, B. C., Ude, A., and Ollero, A. (2016). Extracting objects for aerial manipulation on uavs using low cost stereo sensors. *Sensors*, 16(5):700.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Sánchez-López, J., Fernandez, R. A. S., Bavle, H., Sampedro, C., Molina, M., Pestana, J., and Campoy, P. (2016). Aerostack: An architecture and open-source software framework for aerial robotics. *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 332–341.

Siouris, G. M. (2004). *Missile guidance and control systems*. Springer Science & Business Media.

Spurný, V., Báča, T., Saska, M., Pěnička, R., Krajník, T., Thomas, J., Thakur, D., Loianno, G., and Kumar, V. (2019). Cooperative autonomous search, grasping, and delivering in a treasure hunt scenario by a team of unmanned aerial vehicles. *Journal of Field Robotics*, 36(1):125–148.

Stepanyan, V. and Hovakimyan, N. (2007). Adaptive disturbance rejection controller for visual tracking of a maneuvering target. *Journal of guidance, control, and dynamics*, 30(4):1090–1106.

Strydom, R., Thurrowgood, S., Denuelle, A., and Srinivasan, M. V. (2015). Uav guidance: a stereo-based technique for interception of stationary or moving targets. In *Conference towards autonomous robotic systems*, pages 258–269. Springer.

Suarez, A., Heredia, G., and Ollero, A. (2016). Lightweight compliant arm with compliant finger for aerial manipulation and inspection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4449–4454. IEEE.

Suarez, A., Heredia, G., and Ollero, A. (2019). Compliant aerial manipulators with dual arms. In *Aerial Robotic Manipulation*, pages 83–95. Springer.

Suarez, A., Vega, V. M., Fernandez, M., Heredia, G., and Ollero, A. (2020). Benchmarks for aerial manipulation. *IEEE Robotics and Automation Letters*, 5(2):2650–2657.

Thomas, J., Loianno, G., Sreenath, K., and Kumar, V. (2014). Toward image based visual servoing for aerial grasping and perching. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2113–2118. IEEE.

Tian, Y., Li, Y., and Ren, Z. (2011). Vision-based adaptive guidance law for intercepting a manoeuvring target. *IET control theory & applications*, 5(3):421–428.

Tony, L. A., Jana, S., Bhise, A. A., Varun, V., Mozhi Varman S, A., Vidyadhara, B., Gadde, M. S., Krishnapuram, R., and Ghose, D. (February, 2020). Vision based target interception using aerial manipulation. In *Proceedings of Mohamed Bin Zayed International Robotics Challenge Symposium (MBZIRCS-2020)*. Khalifa University.

Tony, L. A., Jana, S., Bhise, A. A., Varun, V., Mozhi Varman S., A., Vidyadhara, B., Gadde, M. S., Krishnapuram, R., and Ghose, D. (February, 2022). Vision based interception of tethered and swaying targets using aerial manipulation. In *4th International Conference on Communication and Computational Technologies (ICCCT-2022)*. SCRS (accepted for publication).

Tony, L. A., Jana, S., Varun, V., Shorewala, S., Vidyadhara, B., Gadde, M. S., Kashyap, A., Ravichandran, R., and Ghose, D. (September, 2021). UAV collaboration for autonomous target capture. In *Proceedings of 2nd Congress on Intelligent Systems (CIS 2021)*. SCRS.

Triharminto, H. H., Prabuwono, A. S., Adji, T. B., and Setiawan, N. A. (2013). Adaptive dynamic path planning algorithm for interception of a moving target. *International Journal of Mobile Computing and Multimedia Communications (IJMCMC)*, 5(3):19–33.

Tsukagoshi, H., Watanabe, M., Hamada, T., Ashlih, D., and Iizuka, R. (2015). Aerial manipulator with perching and door-opening capability. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4663–4668. IEEE.

Vidyadhara, B., Tony, L. A., Gadde, M. S., Jana, S., Bhise, A. A., Sundaram, S., Ghose, D., et al. (2021a). Design iterations for passive aerial manipulator. *arXiv preprint arXiv:2102.08306*.

Vidyadhara, B., Tony, L. A., Gadde, M. S., Jana, S., Varun, V., Bhise, A. A., Sundaram, S., and Ghose, D. (2021b). Design and integration of a drone based passive manipulator for capturing flying targets. *Robotica*, pages 1–16.

Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649.

Xiong, F., Gou, M., Camps, O., and Sznaier, M. (2014). Person re-identification using kernel-based metric learning methods. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 1–16, Cham. Springer International Publishing.

Zarchan, P. (2012). *Tactical and strategic missile guidance.* American Institute of Aeronautics and Astronautics, Inc.

Zhang, T., Xu, C., and Yang, M. (2017). Multi-task correlation particle filter for robust object tracking. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4819–4827.

Zhang, X., Wang, Y., and Fang, Y. (2016). Vision-based moving target interception with a mobile robot based on motion prediction and online planning. In *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 17–21. IEEE.

Zhao, R., Ouyang, W., and Wang, X. (2013). Person re-identification by salience matching. In *2013 IEEE International Conference on Computer Vision*, pages 2528–2535.

How to cite this article: Tony, L. A., Jana, S., Varun V. P., Bhise, A. A., Varman S., A. M., Vidyadhara B. V., Gadde, M. S., Krishnapuram, R., & Ghose, D. (2022). Autonomous cooperative multi-vehicle system for interception of aerial and stationary targets. *Field Robotics, 2*, 107–146.