Robust Precision Landing for Autonomous Drones Combining Vision-based and Infrared Sensors

Giannis Badakis, Manos Koutsoubelias and Spyros Lalis
Electrical and Computer Engineering Department
University of Thessaly
Volos, Greece
{badakis,emkouts,lalis}@uth.gr

***Abstract*—One of the challenges in drone-based systems is to support automated landing with a high precision that goes beyond the accuracy of standard off-the-shelf GPS. Various efforts have been made to support this, mainly using vision-based and infrared sensors. However, using a single sensor inevitably introduces a single point of failure. To address this problem, we combine a vision-based sensor that detects special visual markers with a sensor that tracks an infrared beacon. We also support a more cautious landing approach for the case where these sensors temporarily fail to detect their targets. We implement our solution in the context of a mature autopilot framework, through modular extensions that are transparent to the rest of the software stack. We evaluate these mechanisms by conducting field experiments using a custom drone, activating faults in the individual precision landing sensor subsystems in a controlled way through interactive commands that are sent to the drone at runtime. The results show that our solution achieves robust precision landing under different failure scenarios while maintaining the accuracy of fault-free sensor operation.**

***Index Terms*—drones, autopilot, precision landing, marker detection, fault-tolerance, ArduPilot, IRLock**

I. Introduction

Multi-rotor unmanned aerial vehicles (drones) are rapidly growing in popularity, playing an increasingly important role in several applications such as courier and delivery services, structural health monitoring search and rescue operations and surveillance. What drives their widespread use is the fact that drones can pilot themselves in a highly autonomous manner. This makes it possible even for inexperienced persons to fly a drone, simply by issuing high-level commands. In the same way, a drone can be flown by a computer program, with little or no human involvement. This opens the way to a new class of automated drone-based systems.

In order to move towards full automation, the drone must be able to take-off and land autonomously. Take-off is typically a straightforward process. Landing, however, can be more tricky. Even though the drone can navigate based on its on-board GPS, the accuracy is typically within a 1-2 meters [1]. This is typically unacceptable if the drone needs to land on a hangar platform or a battery recharging pad.

This research has been co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH - CREATE - INNOVATE, project PV-Auto-Scout, code T1EDK-02435.

While different sensing technologies have been developed to support landing with high precision, the usual approach is to pick one and integrate it in the drone platform. However, this inevitably introduces a single point of failure. In this work, we increase the robustness of precision landing, making it possible to tolerate failures of individual sensor subsystems, which may occur due to the limitations of a particular sensing technology or mere hardware malfunctions of the respective subsystem.

The main contributions of this work are: (i) We develop a new precision landing sensor subsystem, using a low-cost on-board RGB camera combined with software that detects a special visual marker on the landing zone. (ii) We aggregate the values produced by the marker sensor with those of a mature infrared precision landing sensor, to tolerate a single independent failure of any one of the two sensor subsystems. (iii) This extra functionality is smoothly integrated in the popular open-source ArduPilot framework [2], which is widely used in numerous drone platforms. (iv) We introduce a more cautious precision land mode to handle the case where both sensors fail to detect their targets at the same time. (v) The improved precision landing capability is extensively evaluated via field tests using a custom drone. Our results show that our mechanisms provide increased robustness for a range of failure scenarios, and that the marker sensor achieves comparable accuracy to the infrared sensor under normal light conditions.

The rest of the paper is structured as follows. Section II gives an overview of indicative related work. Section III describes our implementation and integration with the ArduPilot framework. Section IV presents an evaluation of our mechanism, focusing on indicative results obtained in our field experiments. Finally, Section V concludes the thesis and points to directions for future work.

II. Related Work

A large body of work investigates methods for detecting pre-defined shapes using a camera at the bottom of the UAV. In [3], autonomous landing on a ship is achieved using a line segment processing technique to detect a standard "H" mark on the landing deck. The same mark is used in [4] with an additional circle around it to enable detection of the landing area from larger distances/heights. The work in [5] proposes a mark consisting of two rings and two upside-down triangles that can be detected from different altitudes, while [6] uses two

 DOI: 10.1109/SAS51076.2021.9530091

adjacent diagonally-placed squares of different size to provide full 3D position estimation, roll, pitch and yaw. A custom stripped pattern is tracked in [7] and [8] using an optical flow sensor along with an IMU and an altimeter to support precision landing on stationary and moving landing pads, respectively.

Another popular approach is to use special visual markers. In [9], an on-board camera is used to determine the 3D position and orientation between the UAV and an AprilTag, guiding the drone accordingly. [10] proposes a custom ArUco marker, which consists of a small marker nested inside a bigger one, and validates the approach through real flight tests for a landing pad mounted on a moving vehicle. The work in [11] employs two AprilTags placed at a known distance from each other to detect the landing target from different altitudes. A more elaborate design is presented in [12], featuring multiple ArUco markers in different sizes surrounded by a circle with another nested marker at the center, leading to improved detection from both high altitudes and very low heights.

Technologies based on infrared LEDs and corresponding sensors have also been successfully used for precision landing. The work in [13] tracks four IR beacons arranged in a "T" pattern to extract the 3D position and yaw of the UAV. A similar approach is followed in [14], using four beacons with a different placement. [15] employs the IRLock sensor to track a MarkOne beacon, together with a rangefinder that returns the vertical distance to the landing position. A more complex IR source is used in [16], consisting of 2×72 LEDs, detected by the drone via a Raspberry Pi NoIR camera.

Some work uses both visual and infrared technologies. For instance, [17] combines a visual "H" mark with 4 IR LEDs at its corners. The latter are used to detect the landing position from large distances using the front-view camera of the UAV, while the former is used in the final phase of the landing process once the image appears in the field-of-view of another camera mounted at the bottom of the vehicle. A special landing pad is presented in [18], featuring an ArUco marker with a translucent white acrylic surface back-lit by an array of 264 IR LEDs. This makes the marker visible for a conventional RGB camera (without an IR filter) even at poor light conditions.

The vast majority of the approaches proposed to support precision landing of UAVs rely on a single sensor. Also, work that combines different sensors typically uses them interchangeably, depending on weather conditions or the phase of the landing process. In our work, we use two different sensors concurrently and combine their information in order to tolerate independent failures of any single sensor subsystem.

III. Implementation

A. The ArduPilot framework

Our implementation is based on the open-source ArduPilot [2] framework (APM), a popular autopilot system used in multi-copters, helicopters, rovers and other vehicles. The software architecture of ArduPilot is shown in Figure 1. Among other basic functions, the core libraries provide support for attitude and position estimation using an Extended Kalman Filter (EKF), precision landing control and motor control.

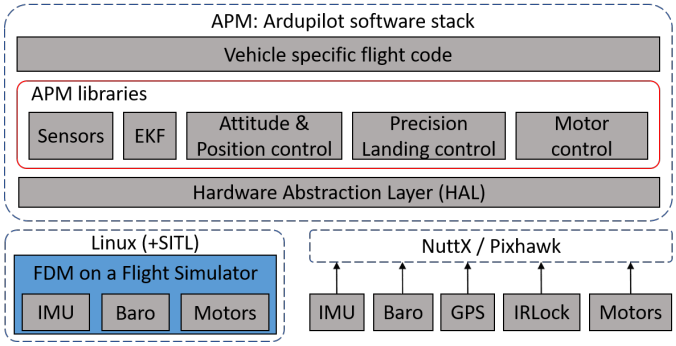


Fig. 1: Software structure of the ArduPilot framework.

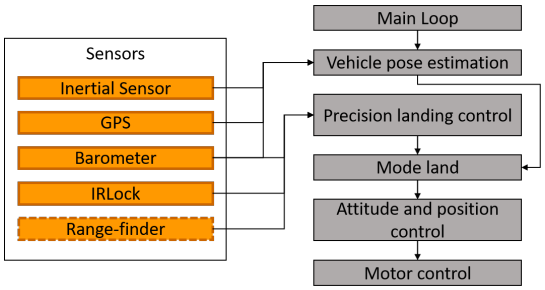


Fig. 2: ArduPilot control flow with precision landing.

Each sensor is accessed via its front-end driver through an interface that consists of several function calls. In turn, the front-end is responsible for invoking the back-end(s) in order to retrieve the data produced by the respective physical sensor(s). There are libraries that implement a variety of such drivers, e.g., for the IMU, the barometer, the GPS, the IRLock subsystem and the motors of the vehicle. Notably, ArduPilot is portable to a large number of platforms and also supports a special software-in-the-loop (SITL) configuration.

The main control loop of ArduPilot when precision landing is enabled is illustrated in Figure 2. The IMU, GPS and barometer sensors are used to calculate the vehicle's current position/attitude, while the IRLock, barometer and range-finder sensors are used to calculate the vehicle's position relative to the landing target. This information is then used to drive the precision landing of the vehicle by issuing commands to its motors/actuators in order to adjust the vehicle's horizontal and vertical position as needed. The loop runs periodically at a frequency of 400 Hz. In case the precision landing sensor fails to provide a valid value, the landing approach continues based only on the estimated drone position.

ArduPilot already provides support for the IRLock target tracking mechanism [19]. IRLock is based on the Pixy camera sensor [20] with special firmware that tracks and reports the position of LEDs at 50 Hz. IRLock can be used in combination with the MarkOne beacon to provide reliable detection in practically all lighting conditions with a range up to 15 meters. Note that IRLock only provides horizontal displacement information, thus the ground distance has to be provided by a barometer or a range-finder.

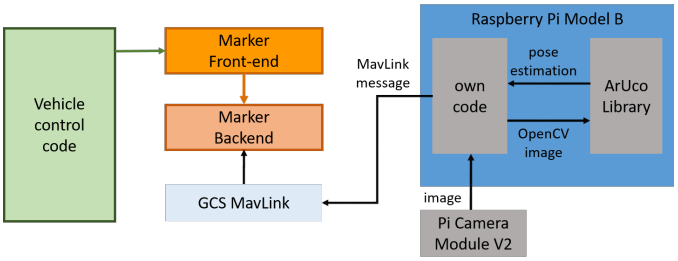


Fig. 3: Design of the marker sensor subsystem.

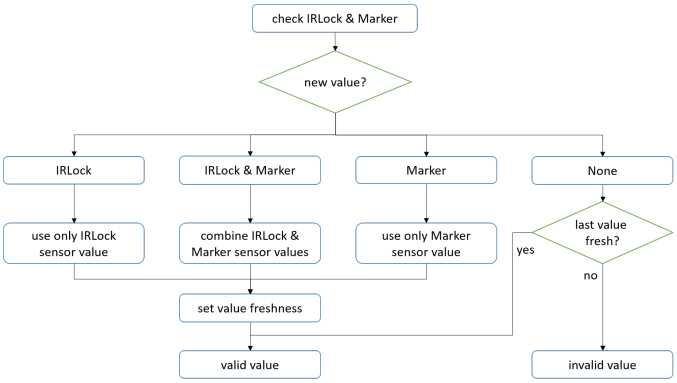


Fig. 4: Sensor aggregation logic.

B. Marker sensor subsystem

Visual markers are often used in robotics to support autonomous navigation. In our work, we employ fiducial markers [21] with a black border and an inner region that encodes a binary pattern. Marker detection is done using the ArUco library [22] on top of OpenCV [23]. ArUco also supports the detection of fractal markers [24].

The marker sensor subsystem is implemented using an on-board System-on-Chip (SoC), based on Raspberry Pi (RPi) model B with a low-cost camera module v2 configured at a resolution of 640×480 (480p) with a rate of 30 fps. The RPi runs the ArUco library to detect the marker in the frames captured by the camera. When the marker is detected in a new frame, the estimated camera position relative to the center of the marker is calculated and made available to the autopilot. The frame capture and processing delay is about 50 ms.

The new sensor subsystem is integrated into the ArduPilot framework following the standard front-end driver and back-end scheme, as shown in Figure 3. The marker detection SoC sends the position estimation information to the back-end of the marker sensor subsystem over a serial connection via the MAVLink protocol [25] using a special message we introduce for this purpose. Each time the back-end receives such a message, it calculates the vehicle's relative position and distance to the center of the marker. The main control logic of the autopilot retrieves this information from the corresponding front-end. Notably, the marker sensor also provides an estimate for the distance of the vehicle to the center of the marker, thus it can be used without an extra sensor for altitude information.

C. Sensor aggregation

To tolerate individual failures of the IRLock and marker sensor, we aggregate the input received from both subsystems by introducing a meta-sensor, which is accessed by the main control logic of the autopilot as usual, through a front-end driver. In turn, the front-end driver invokes the IRLock and marker sensor back-ends to obtain the corresponding position information, which is then combined to provide the information sent to the autopilot.

The aggregation logic of the front-end is shown in Figure 4. If both sensors produce a new value, the two values are combined by calculating their weighted average. The weights are configurable in order to support flexible experimentation. Aggregation is done only for the horizontal position as IRLock does not provide any ground distance information. If only one of the sensors produces a new value, the front-end returns this value to the autopilot. Finally, in case no sensor produces a new value, the front-end returns the last value, provided this is sufficiently fresh, else, it reports that it cannot provide a valid measurement. To deal with the latter case in a more graceful manner, we introduce an additional extension to the autopilot, discussed in the following.

D. Cautious land mode

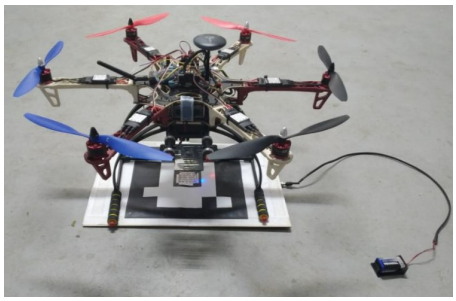
The default operation of ArduPilot in the precision landing mode is to continue the descend even when the precision landing sensor fails to provide fresh/valid measurements. To improve robustness, we introduce a more cautious precision landing mode.

In a nutshell, if the precision landing sensor does not report new/valid measurements, the vertical controller of the autopilot is instructed to pause the descent for a short amount of time. During this period, the sensor is polled to check if the landing target is detected. If so, the landing procedure continues as usual. Else, the controller is instructed to return either to the last altitude where the target was detected successfully or (if this fails too) to a back-off altitude from where a fresh landing attempt is started. The pause period, the back-off altitude and the number of landing attempts before reverting to the default mode, are all configurable.

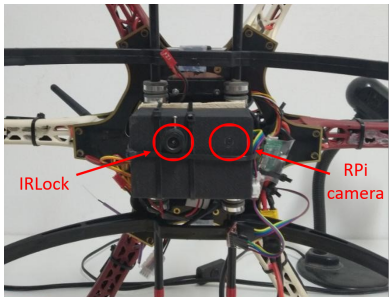
E. Artificial failures

To enable testing in a flexible and reproducible way, we develop a simple mechanism for introducing artificial failures to the individual IRLock and marker sensor subsystems. This is done by modifying the front-end drivers to drop the values received from the corresponding back-ends.

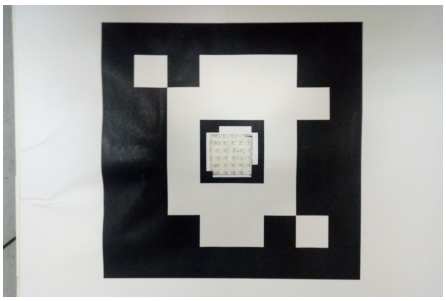
We support two failure modes: *random drop* where a newly acquired value is dropped with some probability, and *systematic drop* where an entire sequence of newly acquired values is dropped. Using these two modes, one can simulate both sporadic and temporary failures of any sensor subsystem. The failure mode and its parameters can be set/changed in a flexible way at runtime, while the drone is in the air, via special MAVLink messages sent to the drone from a ground station.



(a) Drone on the landing target.



(b) Camera sensors (bottom view).



(c) Fractal marker with nested IR beacon.

Fig. 5: Custom hexacopter drone and landing pad used in the field experiments.

IV. Evaluation

We have tested our implementation via extensive simulations, using Gazebo [26] and the official software-in-the-loop (SITL) configuration of ArduPilot [27]. To support marker detection in this setup, we developed a special marker object, extended the default drone model with a second camera and integrated the marker detection component via ROS [28].

Moreover, we have confirmed the smooth operation of our implementation and have evaluated its performance via several experiments in the field, using a drone equipped with the IRLock and marker sensor subsystems. This section focuses on these field trials. Next, we describe the experimental setup and discuss the results that were obtained in different experiments.

A. Hardware setup

The drone used in the field experiments is a custom hexacopter, shown in Figure 5a. The autopilot runs on the CUAV V5 Nano flight controller, which is based on the Pixhawk FMUv5 platform, designed by CUAV in collaboration with the PX4 team. The drone also features the Neo v2 GPS/Compass sensor and an on-board Raspberry Pi (RPi) model B, which communicates with the autopilot board via the MAVLink protocol on top of a serial UART interface.

Figure 5b shows the downwards facing IRLock sensor and the RPi camera module v2 used for marker detection. IRLock generates data at 50 Hz sent to the autopilot board via I2C. The RPi camera connected to the RPi is configured to generate 480p images at a rate of 30 fps.

The landing pad, shown in Figure 5c, consists of a fractal ArUco marker and a MarkOne beacon placed inside the white part of the nested marker. The beacon's surface is covered with a white duct tape (except the IR LEDs) in order for the inner part of the nested marker to remain white and be correctly detected by the marker detection software.

B. Test mission

In all experiments, the drone follows a standard mission specified using DroneKit [29]. An illustration of the mission is given in Figure 6 showing all major positions and waypoints. The photo shows the site where all tests where performed (flat open space with no obstacles).

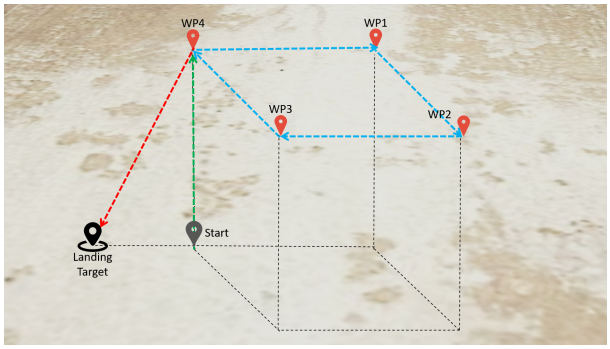
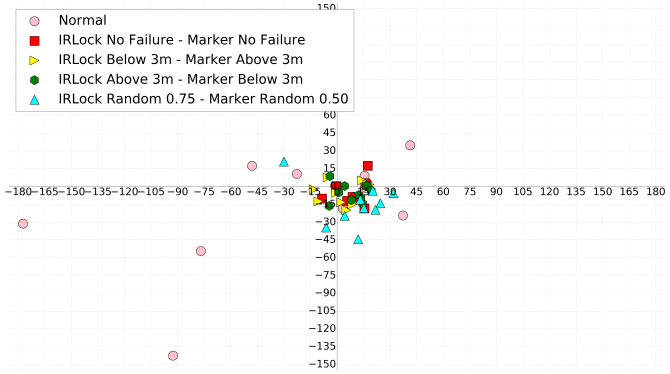


Fig. 6: Mission plan for the experiments.

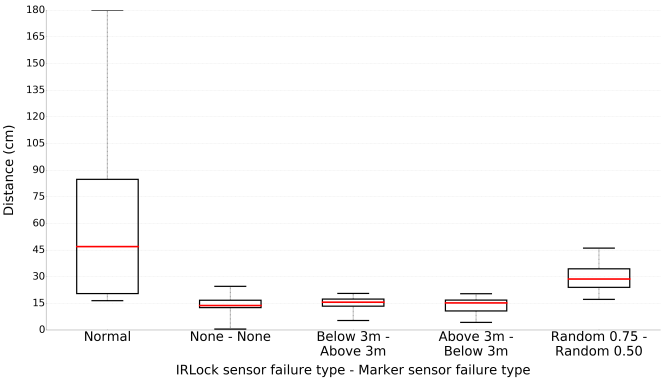
The drone is placed at a start position, 2 meters away from the landing pad. It is then armed and instructed to take off at a certain altitude (green line). When the drone reaches the target altitude, it marks its current GPS position and then follows a horizontal square path (blue line), moving away from the landing target and then returning back to the recorded GPS position. The path is defined via four waypoints (WP1, WP2, WP3, WP4) with the last one being the position recorded after take-off. To increase the number of runs that can be performed without recharging, the distance between two consecutive waypoints is set to 2 meters. When the drone reaches the last waypoint, it starts the landing approach. If a precision landing mode is enabled, the drone tries to detect and land on the center of the landing target (red line). Else, the drone performs a normal landing approach towards the initial start position (along the green line).

C. Experimental results

In a first set of experiments, we evaluate our sensor aggregation approach for different failures of the IRLock and marker sensors. We exploit the developed support for artificial fault injection to implement the following scenarios: (i) The IRLock sensor fails systematically once the drone drops to less than 3 meters from the ground, while the marker sensor fails systematically as long as the drone is above 3 meters. (ii) The IRLock sensor fails systematically as long as the drone is above 3 meters from the ground, while the marker sensor fails systematically once the drone drops to less than

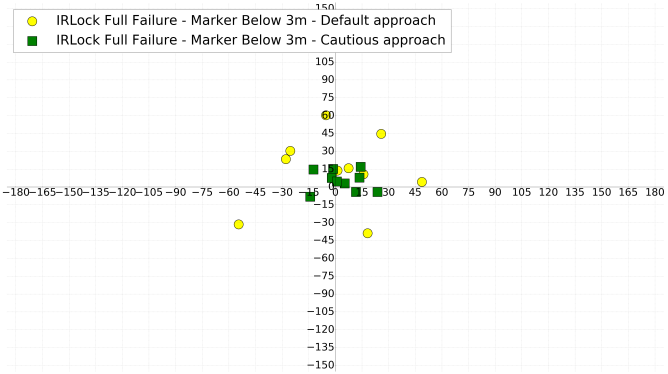


(a) Landing positions relative to target center (units in cm).

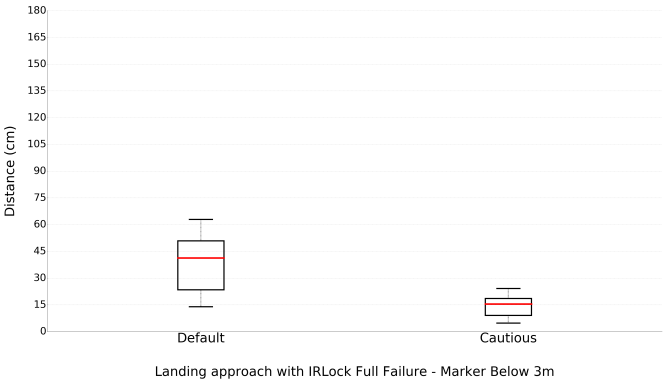


(b) Landing error / distance from target center.

Fig. 7: Sensor aggregation under different failure scenarios.



(a) Landing positions relative to target center (units in cm).



(b) Landing error / distance from target center.

Fig. 8: Cautious vs default precision landing mode.

3 meters. (iii) Both sensors exhibit random failures, with a probability of 0.75 for the IRLock sensor and 0.50 for the marker sensor (which has a lower sampling rate). Note that in this case, both sensors may occasionally fail at the same time. In all experiments, the weights for the sensor aggregation are set to favor the readings of the IRLock sensor over those of the marker sensor whenever both deliver valid measurements.

As a reference, we report the results achieved using the normal land mode where the drone does not employ any precision landing sensor. This is equivalent to the scenario where both sensors fail systematically during the entire landing approach. We also show results for the ideal scenario where both sensors function properly during the entire landing approach. To minimize side-effects due to external factors, all experiments were performed under normal daylight and light wind conditions with only a few moderate gusts. We wish to note that the purpose of these experiments is to evaluate the ability of our mechanisms to increase the robustness of precision landing in case of sensor failures, rather than finding the limitations of each individual sensor and the conditions under which it will indeed fail to provide valid measurements (such an exploration is beyond the scope of this paper).

Each scenario was tested 10 times. In each experiment, we record the landing position of the drone and measure the landing error, i.e., the distance from the center of the landing pad. For the normal land mode, the landing error is measured with respect to the initial take-off position (in this case, the drone is not even aware of the landing pad).

The results are illustrated in Figure 7. When using the normal land mode, as expected, the result is heavily affected by the GPS error [1], with an average landing error of 67.15 cm and a worst case of 180 cm away from the initial position. Obviously, this is unacceptable if the drone is required to land inside a limited area or on a small platform.

The landing accuracy improves substantially when using the aggregation meta-sensor, even when the individual physical sensors experience systematic failures. As long as the sensors do not fail at the same time (IRLock Below 3m - Marker Above 3m / IRLock Above 3m - Marker Below 3m), the meta-sensor masks the failures and the drone lands with the same accuracy as when both sensors function properly (IRLock None - Marker None). Note that the marker sensor guides the vehicle equally well to the IRLock sensor. In all the above cases, the average error is about 15 cm and the worst case error is roughly 25 cm, an order of magnitude lower than landing without having a (working) precision landing sensor.

The landing accuracy decreases in the scenario where both sensors may fail at the same time (IRLock Random 0.75 - Marker Random 0.50). Since these double failures cannot be masked by the meta-sensor, they have a visible impact on the

landing accuracy. More specifically, the average error jumps to about 30 cm and the worst case to more than 46 cm. This is a deterioration of roughly $2x$ compared to the scenarios where at least one of the sensors works properly and the meta-sensor is able to provide the autopilot with reliable measurements during the entire landing approach. This further stresses the importance of being able to tolerate single independent failures of any of the underlying physical precision landing sensors.

In a second set of experiments, we evaluate the cautious landing mode where the landing attempt is repeated if the precision landing sensor fails to provide measurements during descent. In these trials, we turn-off the IR beacon hence the IRLock sensor fails systematically. We additionally introduce an artificial failure in the marker sensor when the drone is below 3 meters from the ground, so the aggregation meta-sensor also fails to provide valid values. The proper operation of the marker sensor is restored during the second landing attempt, allowing the meta-sensor to deliver valid measurements. As a reference, we use the default mode where the autopilot continues the descent during the first landing attempt even though the precision landing sensor stops working properly.

Each test is repeated 10 times, under conditions similar to those in the first set of experiments. The results are summarized in Figure 8. As can be seen, the accuracy achieved by the cautious mode is practically identical to that achieved in the previous scenarios where at least one of the precision landing sensors functions properly during the initial landing attempt. In contrast, when using the default mode, the drone lands with a much larger error, more than $2x$. While the more cautious approach increases the landing time from 17.5 to 41.5 seconds on average, this extra delay will usually be more than welcome in return for a more accurate landing, provided the drone does not need to perform a radical emergency landing.

V. Conclusions

We have extended the precision landing capability of the popular ArduPilot by implementing a new sensor subsystem for the detection of fiducial markers, by adding a meta-sensor that aggregates the readings of the IRLock and marker sensor in a transparent way for the rest of the autopilot framework, and by introducing a more cautious land mode. Our implementation was tested extensively via simulations, and was evaluated in the field by injecting artificial faults in the individual sensor subsystems in a controlled way. Our results show that our mechanism achieves the desired tolerance to the failures of any single sensor subsystem at any point in time, while maintaining good accuracy that is an order of magnitude better than the normal land mode performed in the absence of a working precision landing sensor.

Our work opens-up the way for the smooth integration and experimentation with additional precision landing sensor mechanisms, which can be based on completely different technologies, such as magnetic materials or ultrasound signals. Such technologies could be easily combined with the current sensor subsystems to further improve the landing accuracy for applications that have even stricter requirements.

References

[1] "Gps accuracy," www.gps.gov/systems/gps/performance/accuracy.
[2] "Ardupilot autopilot software suite," https://ardupilot.org/ardupilot.
[3] S. Lin, M. Garratt, and A. Lambert, "Monocular vision-based real-time target recognition and tracking for autonomously landing an uav in a cluttered shipboard," *Autonomous Robots*, vol. 41, no. 4, pp. 881–901, 2017.
[4] C. Patruno, M. Nitti, A. Petitti, E. Stella, and T. D'Orazio, "A vision-based approach for unmanned aerial vehicle landing," *Journal of Intelligent & Robotic Systems*, vol. 95, no. 2, pp. 645–664, 2019.
[5] F. Cocchioni, V. Pierfelice, A. Benini, A. Mancini, E. Frontoni, P. Zingaretti, G. Ippoliti, and S. Longhi, "Unmanned ground and aerial vehicles in extended range indoor and outdoor missions," in *Proc. Intl Conference on Unmanned Aircraft Systems*, 2014, pp. 374–382.
[6] L. Wei, W. J. Dong, and L. M. Yang, "A vision-based attitude/position estimation for the automatic landing of unmanned helicopter on ship," in *Proc. Intl Conference on Watermarking and Image Processing*, 2017, pp. 1–5.
[7] M. Al-Sharman, B. Emran, M. Jaradat, H. Najjaran, R. Al-Husari, and Y. Zweiri, "Precision landing using an adaptive fuzzy multi-sensor data fusion architecture," *Applied Soft Computing*, vol. 69, pp. 149–164, 2018.
[8] B. Herissé, T. Hamel, R. Mahony, and F. Russotto, "Landing a vtol unmanned aerial vehicle on a moving platform using optical flow," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 77–89, 2012.
[9] G. Wang, Z. Liu, and X. Wang, "Uav autonomous landing using visual servo control based on aerostack," in *Proc. Int Conference on Computer Science and Application Engineering*, 2019, pp. 1–6.
[10] J. S. Wynn and T. W. McLain, "Visual servoing with feed-forward for precision shipboard landing of an autonomous multirotor," in *Proc. American Control Conference*, 2019, pp. 3928–3935.
[11] Z. Li, Y. Chen, H. Lu, H. Wu, and L. Cheng, "Uav autonomous landing technology based on apriltags vision positioning algorithm," in *Proc. Chinese Control Conference*, 2019, pp. 8148–8153.
[12] X. Liu, S. Zhang, T. Jiayi, and L. Longbin, "An onboard vision-based system for autonomous landing of a low-cost quadrotor on a novel landing pad," *Sensors*, vol. 19, no. 21, pp. 4703–4722, 2019.
[13] K. Wenzel, P. Rosset, and A. Zell, "Low-cost visual tracking of a landing place and hovering flight control with a microcontroller," *Journal of Intelligent & Robotic Systems*, vol. 57, no. article 297, 2010.
[14] K. Wenzel, A. Masselli, and A. Zell, "Automatic take off, tracking and landing of a miniature uav on a moving carrier vehicle," *Journal of Intelligent & Robotic Systems*, vol. 61, pp. 221–238, 2011.
[15] M. R. Hayajneh and A. R. E. Badawi, "Automatic uav wireless charging over solar vehicle to enable frequent flight missions," in *Proc. Intl Conference on Automation, Control and Robots*, 2019, pp. 44–49.
[16] E. Nowak, K. Gupta, and H. Najjaran, "Development of a plug-and-play infrared landing system for multirotor unmanned aerial vehicles," in *Proc. Conference on Computer and Robot Vision*, 2017, pp. 256–260.
[17] L. Wang and X. Bai, "Quadrotor autonomous approaching and landing on a vessel deck," *Journal of Intelligent & Robotic Systems*, vol. 92, pp. 125–143, 2018.
[18] J. S. Wynn and T. W. McLain, "Visual servoing for multirotor precision landing in daylight and after-dark conditions," in *Proc. Intl Conference on Unmanned Aircraft Systems*, 2019, pp. 1242–1248.
[19] "Irlock target tracking system," https://irlock.com.
[20] "Pixy vision sensor," https://pixycam.com/pixy-cmucam5/.
[21] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and R. Medina-Carnicer, "Generation of fiducial marker dictionaries using mixed integer linear programming," *Pattern Recognition*, vol. 51, pp. 481–491, 2016.
[22] F. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and Vision Computing*, vol. 76, pp. 38–47, 2018.
[23] "Opencv library," https://opencv.org.
[24] F. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Fractal markers: a new approach for long-range marker pose estimation under occlusion," *IEEE Access*, vol. 7, pp. 169 908–169 919, 2019.
[25] "Mavlink developer guide," https://mavlink.io.
[26] "Gazebo open-source 3d robotics simulator," http://gazebosim.org.
[27] "Ardupilot-gazebo plugins," https://github.com/khancyr/ardupilot_gazebo.
[28] "Robot operating system," https://www.ros.org.
[29] "Developer tools for drones," https://dronekit.io.