Received: 29 July 2021 | Revised: 9 January 2022 | Accepted: 29 January 2022

DOI: 10.1002/stc.2942

RESEARCH ARTICLE

WILEY

Autonomous wireless sensor deployment with unmanned aerial vehicles for structural health monitoring applications

Hao Zhou[1] | **Jerome Lynch**[2] | **Dimitrios Zekkos**[3]

[1]Department of Civil and Env. Engineering, University of Michigan, Ann Arbor, Michigan, USA

[2]Department of Civil and Env. Engineering, Duke University, Durham, North Carolina, USA

[3]Department of Civil and Env. Engineering, University of California, Berkeley, California, USA

Correspondence
Jerome Lynch, Department of Civil and Env. Engineering, Duke University, Durham, NC, USA.
Email: jerome.lynch@duke.edu

Funding information
Office of Naval Research, Grant/Award Number: N00014-21-1-2033; National Science Foundation, Grant/Award Numbers: 1362975, 1442773, 1446521, 1831347

Summary

Unmanned aerial vehicles (UAVs) equipped with onboard sensors empower end-users to collect data within a wide range of civil engineering applications such as structural condition assessment. While UAVs have been used primarily as mobile sensing platforms to obtain imagery or other data, their mobility can also be used to deploy sensor networks. In this study, the feasibility of using an autonomous UAV to deploy wireless sensors in structural monitoring applications is proposed. While GPS-based waypoint navigation is available for automating UAV flight operations, this approach does not provide the accuracy necessary for the precision placement of sensor payloads on structures. Computer vision-based pose estimation is proposed to improve the accuracy of UAV localization for sensor placement. Variably sized fiducial markers integrated into a single pattern are applied to the surface of the structure and adopted as navigation and precision landing targets that identify sensor placement locations. Visual and inertial measurements are fused by means of a discrete Kalman filter to further increase the robustness of the relative position estimation algorithm that is included in the proportional-integral-derivative (PID) control law used for UAV landing. Outdoor experiments under realistic operational conditions are conducted to validate the proposed vision-aided control of the UAV for sensor placement; the UAV is able to land on a predefined landing point within 10 cm. A UAV moving a wireless accelerometer to locations on a beam is used to experimentally show the validity of automating UAV sensor placement for modal analysis using reconfigurable sensor network topologies.

KEYWORDS

autonomy, computer vision, Kalman filter, mobility, UAV, wireless sensor

1 | INTRODUCTION

Adoption of sensing is growing increasingly attractive in a wide range of civil engineering applications due to the reduction of sensor cost, the integration of wireless communication that make deployments easier, and the improvement of analytical frameworks that extract value from collected data.[1] This has made monitoring common in many field applications such as structural health monitoring (SHM).[2–5] In SHM applications, dense sensor arrays are often needed which can drive system costs high. For example, long-span bridges could require hundreds of sensors to ensure

sufficient spatial coverage for SHM.[6] If sensors could be moved, the density of permanent sensors could be reduced, thereby lowering costs while rendering systems more flexible to adapt to the needs of the SHM application. To date, the placement of sensors on a structure are done manually with the assumption that they do not move. However, autonomous robotic systems could be developed for the initial placement and later movement of sensors which would make monitoring systems more flexible and cost-effective.

Unmanned aerial vehicles (UAVs) could potentially be one such solution that offer mobility to sensors that allow for the collection of data that are difficult to obtain using conventional stationary monitoring approaches. The data collection capabilities of UAVs have already proven valuable in many civil engineering applications such as infrastructure inspection, traffic monitoring, and remote sensing.[7–9] Computer vision methods offer a promising approach to identifying the condition of infrastructure with inexpensive cameras installed on the UAV. For example, recent studies have demonstrated several innovative applications of UAVs equipped with cameras (e.g., optical, infrared) to conduct infrastructure monitoring such as delamination detection of concrete bridge decks,[10,11] modal analysis of a pedestrian suspension bridge,[12] and visual inspection of steel girder bridges.[13,14] In all of these applications, the UAV systems are primarily used as a mobile data collection platform to observe the system from afar and make no direct contact with the structure. Also, navigation of the UAV is controlled by either a human pilot operating the UAV or using a GPS module providing waypoint coordinates for autonomous flight operations. Both navigation methods lack accuracy and struggle to give desirable or reliable measurements for cases where precise spatial control of the UAV is required. The utility of UAVs could be enhanced if they not only carry sensor payloads, but also have the ability to deploy sensors. This can be especially valuable in applications where data collection may be required over a longer period of time (e.g., days and years) than what current UAV flight endurances allow (e.g., minutes and hours). In the literature, mobile sensor networks deployed by robots in structures has been developed and validated by several researchers. Huston et al. studied the use of a mobile robot that was able to crawl along bridge girders while measuring girder flange thickness with an ultrasonic sensor.[15] Zhu et al. prototyped a climbing robot equipped with magnetic wheels capable of adhering to and navigating on a steel bridge. The robots carried accelerometers as a robot payload and moved around the bridge to sample structural vibrations.[16] In this paper, aerial delivery of sensing payloads based on computer vision and position estimation is proposed as part of an autonomous UAV sensor deployment system. Aerial deployment has advantages over wheel-based robots including more freedom in moving to different locations.

Precision control of a UAV to land on desired positions (i.e., within 50 cm or less) is necessary for effective sensor placement. GPS-based waypoint navigation techniques used in other SHM applications (e.g., collection of imagery data) would be insufficient due to UAV positioning errors being as large as meters that would result in inaccurate and unsuccessful sensor placement. Modern computer vision object detection and pose estimation algorithms are a promising alternative to GPS. Autonomous landing of UAVs using vision as the primary data source is currently an active topic of research. Among early investigations, printed patterns have been used to mark the landing target. Saripalli et al. demonstrated vision-based autonomous landing of a model helicopter on an "H"-shaped pad; landing position accuracy was reported to be within 40 cm.[17] To extend the detection distance, Merz et al. proposed a landing pattern consisting of five concentric circle triplets of different size (with radii varying from 2 to 32 cm) achieving a touch down precision of 42 cm.[18] Lange et al. designed a landing pattern with several concentric white rings on a black background and was able to hover a Hummingbird quadcopter above the pattern with a maximum deviation of 23 cm over 5 min.[19] A drawback of the aforementioned printed patterns is that they do not have an extensible design that limits their usage when multiple landing targets in a structure are required. Also, the detection performance of a UAV using these markers under challenging scenarios such as low lighting has not been rigorously analyzed. To address these challenges, researchers have developed fiducial marker systems with a large number of distinguishable patterns and features (e.g., size) that allow a UAV to robustly estimate its location and pose under challenging field conditions (e.g., ARToolKit,[20] ArUco,[21] and AprilTag[22]). For example, Borowczyk et al. gave a demonstration of autonomous landing of a DJI M100 quadcopter on a moving vehicle (moving up to 50 km/h) using a single 30 cm $\times$ 30 cm AprilTag for visual estimation of the vehicle.[23] Chaves et al. accomplished autonomous landing of a UAV (Parrot AR.Drone) on a Segway using a landing platform with four AprilTags with one large marker in the center for initial detection and three small markers on the side for fine pose control at close range when completing the landing maneuver.[24] The drawback of this system is that computations were not done on the UAV so a separate laptop was needed to remotely execute the control law. To further extend the detection range, Araar et al. designed a landing pad using a total of 28 AprilTags with bigger tags surrounding smaller ones resulting in an 8-cm landing error when landing on a stationary target.[25] Similar to Chaves et al.,[24] all computations were run on a separate laptop computer. These works reveal that a single fiducial

marker does not provide the UAV the range necessary for detection from afar while being sufficiently small for precision navigation at close range. Also, landing pads with too many markers have high computational costs requiring remote computers to run the control law. In this paper, a simple yet universal landing pattern for different detection ranges is proposed for detection and use by a UAV onboard computer in near real-time.

In this study, multirotor UAVs are explored for autonomously deploying wireless sensors for structural monitoring applications. The work emphasizes the integration of precise landing and mission management capabilities within the onboard computer of the UAV for truly autonomous operations. Figure 1 provides the operational principles of the autonomous UAV-based sensor deployment system proposed including the use of fiducial-based landing pads for placement of a wireless sensor that can be moved from location to location. The proposed landing pad design is easily adjustable and able to provide reliable visual estimation by the UAV (using an onboard computer) during the entire landing process, thereby ensuring an accurate placement of the sensor payload. The envisioned applications include movement of sensors (e.g., accelerometers) on a structure for structural monitoring (with locations predetermined and marked with landing pads). The work aims to make three major intellectual contributions. First, a computer vision approach using four AprilTag markers for a single landing pad is created to trade off precision with onboard computational time for real-time control of the UAV landing. Second, a fully autonomous system architecture is advanced to control UAV flight operations and sensor placement using only the onboard computing resources of the UAV. Third, the integrated UAV system is demonstrated to autonomously perform modal analysis of a simply supported beam where the only human intervention is impacting the beam with a modal hammer (which emulates ambient vibrations). This work evaluates the precision and repeatability of the autonomous landing process for sensor placement. The work also showcases the quality of the sensor data collected by performing complete modal analysis of the monitored structure using the reconfigurable sensor networks.

2 | UAV PLATFORMS: HARDWARE AND SOFTWARE

2.1 | UAV hardware

In this study, two UAV platforms are used: a 3D Robotics (3DR) X8 octocopter and a Lumenier QAV210 quadcopter (Figure 2). The X8 aluminum frame is a light and sturdy "X" shape and features two motors spinning in opposite directions on each of the four arms (i.e., eight motors in total). The eight Sunnysky 2206-12 800 Kv motors give the UAV

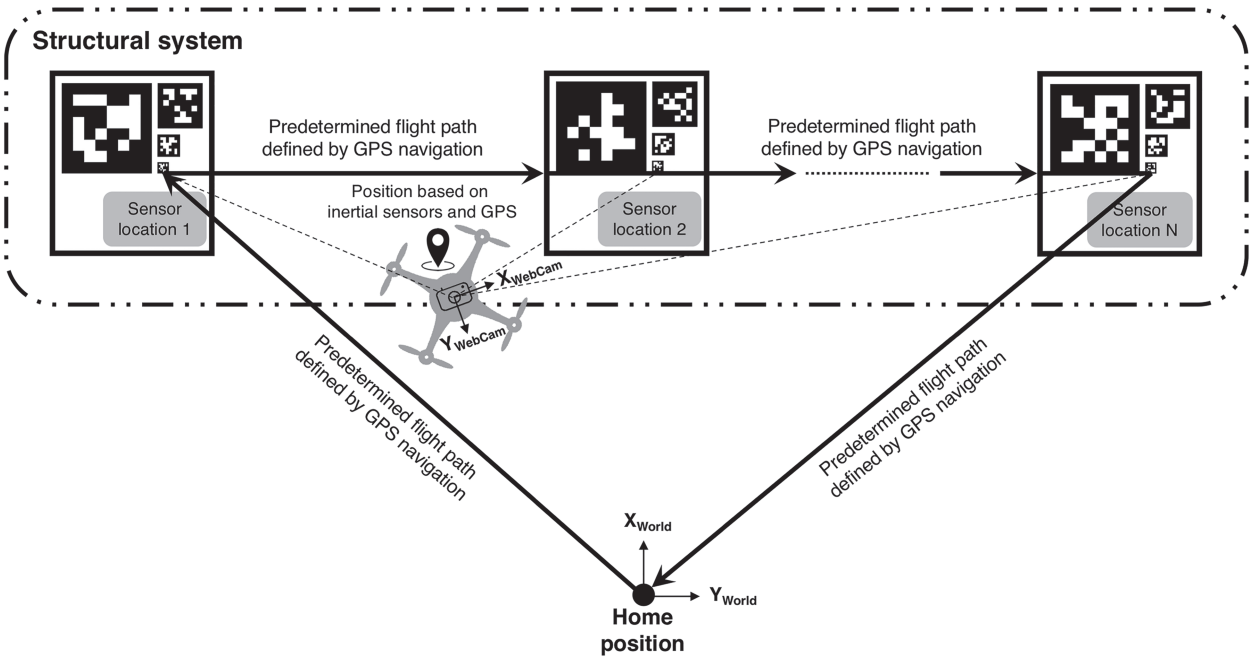
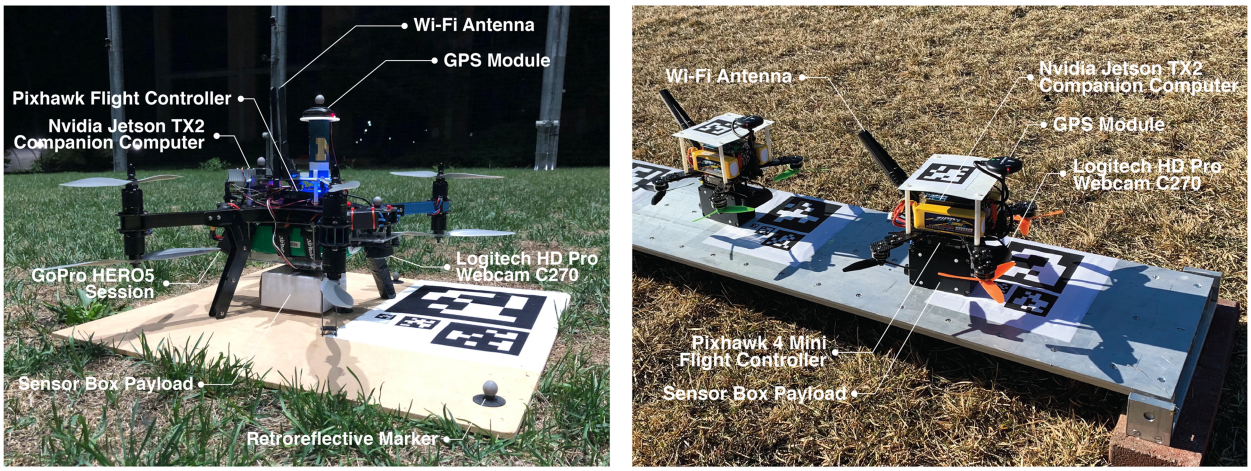


FIGURE 1 Overview of the autonomous UAV system for wireless sensor deployment



(a) (b)

FIGURE 2 Customized multi-rotor UAVs used in this study: (a) the 3DR X8 octocopter sitting on the landing pad with a sensor box attached (Qualisys retroreflective passive markers mounted on UAV and landing pad for pose tracking in M-Air); (b) two Lumenier QAV210 quadcopters carrying wireless sensor payloads on a simply supported aluminum beam (AprilTag markers on beam and UAVs are shown)

capacity to carry nearly a 1-kg payload. The eight motors also provide redundancy offering enough lift and control should a motor burn out. The X8 is powered by a 16,000 mAh 4S 20C LiPo battery and can stay aloft for about 15 min. The QAV210 UAV is smaller with a diagonal length of 210 mm. The QAV210 has a symmetric carbon fiber frame design featuring four efficient Lumenier RX2206-11 2350 Kv motors. The QAV210 has a payload capacity of 300 g and a flight endurance of about 10 min using two 2200 mAh 3S 40C LiPo batteries in parallel. In this study, the heavy-duty X8 is mainly used for thorough testing of the UAV control algorithms with a heavy payload carried on board, while the mini QAV210 is used during the experiments related to delivering light-weight wireless sensor nodes onto a narrow beam to validate system integration and autonomy.

The X8 comes preinstalled with an original 3DR Pixhawk flight controller first released in 2013.[26] The Pixhawk has a 32-bit STM32F427 ARM Cortex-M4 processor with 256 KB RAM and 2 MB Flash, and operates at 168 MHz. The flight controller includes two gyroscope/accelerometer sensors (TDK InvenSense MPU6000 gyroscope/accelerometer, and STMicroelectronics L3GD20H gyroscope/LSM303D accelerometer), a 14-bit STMicroelectronics LSM303D magnetometer, and MEAS MS5611 barometer. The Pixhawk provides many connectivity options including five universal asynchronous receiver/transmitters (UARTs), two controller area network (CAN) ports, and one inter-integrated circuit (I^2C) interface. An external u-blox LEA-6H GPS module is paired with the Pixhawk for outdoor navigation. For the QAV210, its small frame size requires a flight controller with a scaled-down form factor. A Holybro Pixhawk 4 Mini is chosen which has half the footprint of the 3DR Pixhawk but has higher computing performance. The Pixhawk 4 Mini features an upgraded 32-bit STM32F765 Arm Cortex-M7 processor running at 216 MHz with 512-KB RAM and 2-MB memory. The enhanced onboard sensor suite includes an InvenSense ICM-20689 and Bosch BMI055 gyroscope/accelerometer pair, an iSentek IST8310 magnetometer, and the same MS5611 barometer as the 3DR Pixhawk. The external GPS sensor is also upgraded to a u-blox Neo-M8N module. Both flight controllers have a FrSky XSR receiver connected via SBus so that a user can manually command the vehicle using a remote controller (RC) radio transmitter that operates on the 2.4-GHz frequency with an approximate communication range of 1 km.

To expand the onboard computational capabilities of both UAVs, a more powerful single-board computer is integrated. The Nvidia Jetson TX2 is selected as the companion computer to perform tasks on the UAVs that are computationally resource intensive. The TX2 is equipped with a 256-core Pascal graphics processing unit (GPU), a dual-core Nvidia Denver 2.0 central processing unit (CPU), a quad-core ARM Cortex-A57 CPU, and 8 GB 128-bit LPDDR4 memory. In addition, the TX2 includes Wi-Fi communication capabilities with a range of approximately 100 m. A small carrier board (Connect Tech's Orbitty Carrier) to which the TX2 module is attached is selected. This board ($87 \times 50 \times 15$ mm^3) takes little space on the UAV but offers a variety of communication ports (one Universal Serial Bus (USB), two UARTs, one I^2C, and four general-purpose input/outputs [GPIOs]). Communication between the flight controller and the TX2 is established using the UART with a baud rate at 921,600. A ground-based personal computer

(PC) is also used to communicate with the TX2 through its 5-GHz Wi-Fi interface. The flight controller takes commands from the TX2 in the form of MAVLink messages[27] posted over the UART port. At any time, a human pilot can take control of the UAV by commanding the flight controller through the FrSky Taranis X9D transmitter which can communicate up to 1 km line of sight (although this will not be needed in this study). The system architecture is shown in Figure 3.

The camera is another critical component for the precise control of both UAVs. A downward facing Logitech C270 high definition (HD) web camera is connected to the bottom of the UAV and attached directly to the TX2 via USB. This low-end webcam is purposely chosen due to its lack of auto-focus functionality because auto-focus could create blurry images at high speeds. Despite the camera's ability to record 720p HD video clips, image resolution is set to a much lower 640×480 px^2 resolution so that images can be processed in real-time on the TX2. The camera is mounted beneath the front of the UAV using rubber dampers that dampen vibrations. The use of a gimbal is intentionally avoided to ensure an unfiltered view of the ground is obtained from which the pose of the UAV can be accurately estimated. While the QAV210s carry only one camera, an additional camera (GoPro HERO5 Session) is included in the X8 for flight video logging during validation experiments (and not for use in pose estimation during landing) is communicated to a ground PC using 5-GHz Wi-Fi.

The Martlet wireless sensing node[28] developed at the University of Michigan is selected as the primary data collection platform for accelerometers used to measure structural vibrations. The computing core of the Martlet is a 16-bit Texas Instruments (TI) TMS320F28069 modified microcontroller unit (MCU) with a clock frequency up to 80 MHz. The Martlet contains a 9-channel dual sample-and-hold 12-bit analog-to-digital converter (ADC) capable of sampling analog signals at a maximum sampling frequency of 3 MHz. An ADC sensor interface board is attached on top of the Martlet baseboard to provide bandpass filtering and amplification of input analog signals (100 Hz cutoff frequency and 1× gain in this study). Wireless communication between the Martlet and a ground-based PC is established through a power amplified TI CC2520 2.4-GHz IEEE 802.15.4 transceiver integrated with Martlet. Due to being power amplified, the Martlet transceiver can communicate up to a range of approximately 1 km. A customized enclosure for housing the Martlet and a 2-g Crossbow CXL02LF1 accelerometer (1-mg RMS noise floor) is selected. The sensor enclosure is attached to the bottom of the QAV210 and will move together with the vehicle (Figure 2b).

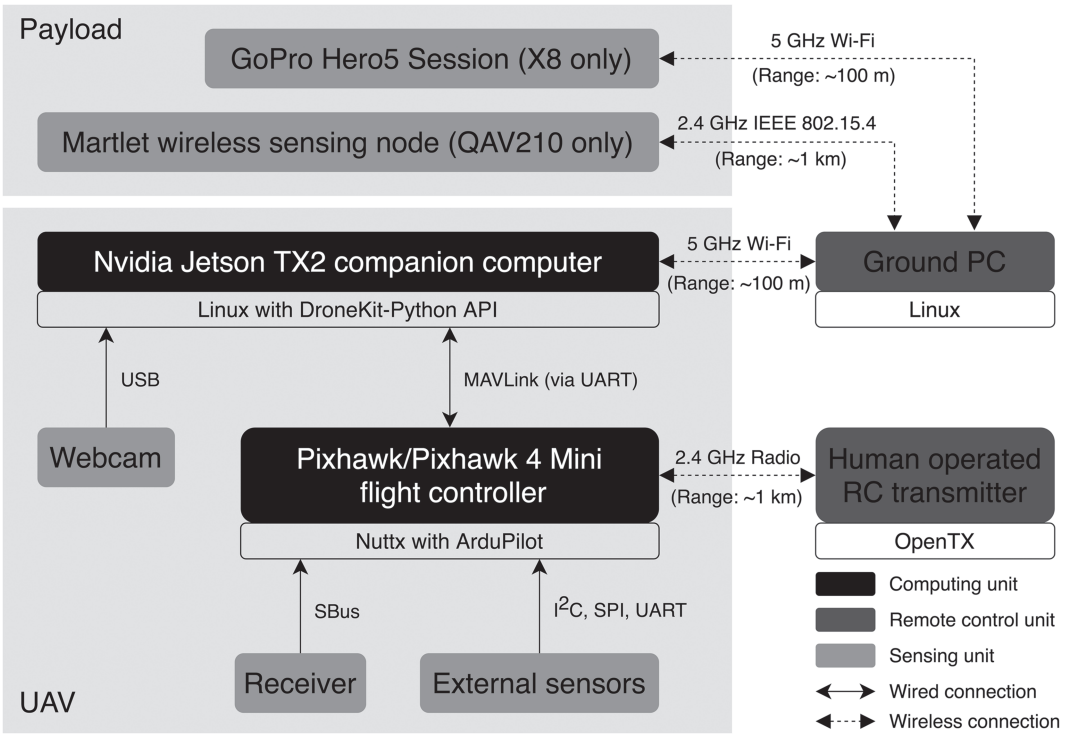


FIGURE 3 UAV system hardware architecture showing components and communication links

2.2 | Embedded software architecture

Embedded software is needed to automate the operations of the UAVs for the deployment of wireless sensors used for monitoring civil engineering systems. Software is written for the two onboard computing elements of the UAV: the TX2 companion computer and the Pixhawk flight controller. High-level flight planning like mission management and compute-intensive tasks like visual pose estimation will be executed using the TX2, while position and attitude control of the UAV are implemented on the flight controller. Figure 4 shows the layout of the software architecture distributed across the two computing elements. To speed up the development of the UAV, the open-source ArduCopter firmware from the ArduPilot project[29] is selected to run on the Pixhawk flight controller as the real-time flight control stack. ArduCopter provides reliable and responsive flight control operations for UAVs in a full range of flight modes including manual and automatic flight operations. High-level control abstractions and interfaces are well documented that enable customized flight features and the development of complex use cases. ArduCopter also has well defined communication interfaces that allow a companion computer (like the TX2) to gain access to flight data and to allow extra power to handle computationally intensive tasks not easily executable on the flight controller.

The main loop (Figure 4) of the ArduCopter flight code includes a 24-state extended Kalman filter (EKF)[30,31] for vehicle state estimation (e.g., vehicle attitude using quaternions which are more computationally efficient than Euler angles, $\vec{q} = [q_0\, q_1\, q_2\, q_3]^T \in \mathbb{R}^4$, vehicle velocity in the global North-East-Down [NED] frame $\vec{v} = [v_N\, v_E\, v_D]^T \in \mathbb{R}^3$, vehicle position in the NED frame $\vec{r} = [r_N\, r_E\, r_D]^T \in \mathbb{R}^3$, gyro bias offsets in the UAV's local body frame $\overrightarrow{b_g} = \left[b_{gX}\, b_{gY}\, b_{gZ}\right]^T \in \mathbb{R}^3$, gyro scale factor in the local IMU frame $\overrightarrow{s_g} = \left[s_{gX}\, s_{gY}\, s_{gZ}\right]^T \in \mathbb{R}^3$, acceleration bias in vehicle body Z direction $b_{aZ} \in \mathbb{R}^1$, earth magnetic field in the NED frame $\overrightarrow{m_e} = [m_N\, m_E\, m_D]^T \in \mathbb{R}^3$, body magnetic field $\overrightarrow{m_b} = [m_X\, m_Y\, m_Z]^T \in \mathbb{R}^3$, and lateral wind velocity $\overrightarrow{v_w} = [v_{wN}\, v_{wE}]^T \in \mathbb{R}^2$). In this study, the vehicle attitude $\vec{q}$, velocity $\vec{v}$, and position $\vec{p}$ will be used for control of the UAV. The EKF is designed to linearize the nonlinear UAV flight dynamics and sensor measurement equations using IMU dead-reckoning to propagate the state and onboard sensor (e.g., GPS and barometer) measurements to update the state estimation. Advantages of the EKF include being able to switch between sensors (in case a sensor fault is identified) and the estimation of external flight variables such as gyro and accelerometer biases, and wind speed leading to better flight performance. The ArduCopter main loop is run on the Pixhawk at 400 Hz including the

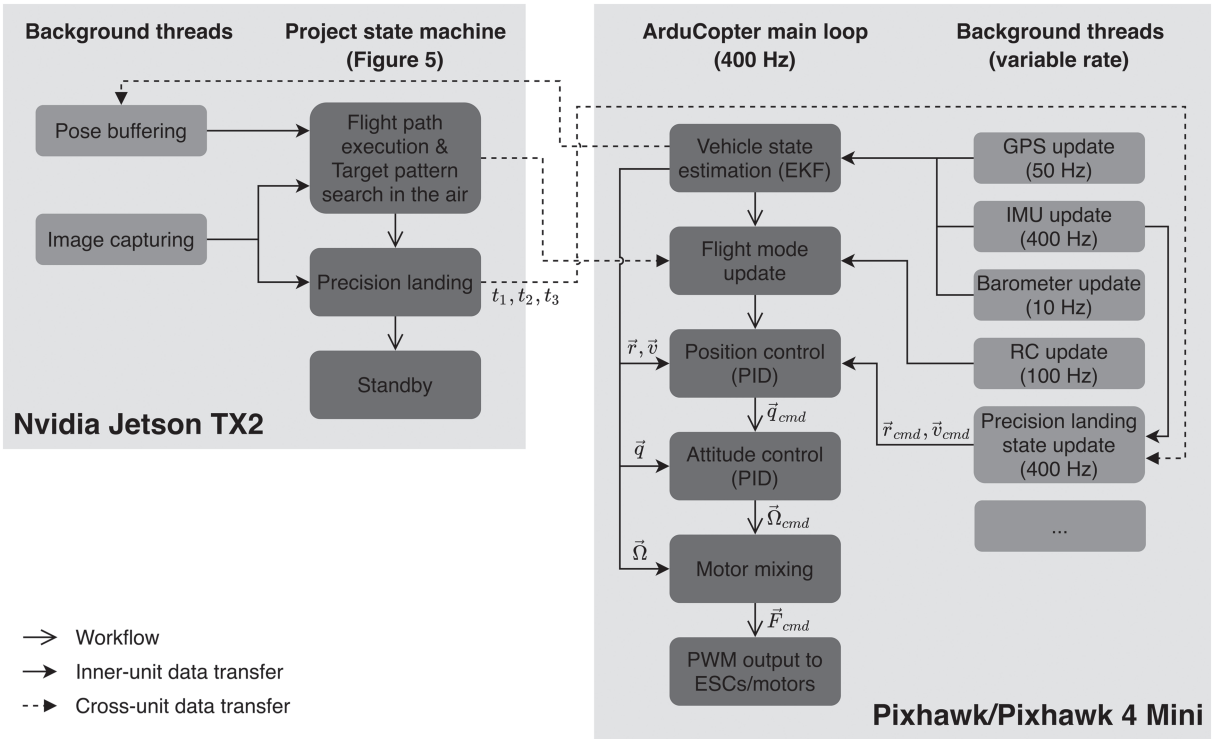


FIGURE 4 Software architecture for the UAV platform where the TX2 provides high-level mission management and image processing while the Pixhawk flight controller is responsible for vehicle state estimation and control

EKF filter. A number of background threads are running constantly on the flight controller to provide input to the EKF algorithm including updates of the GPS (50 Hz), barometer (10 Hz), and IMU (400 Hz). As a result, the EKF provides estimations of the UAV attitude, position, and velocity using available data at the 400 Hz main loop execution rate.

To control UAV motions for autonomous flight, ArduCopter implements a cascaded control structure with a position controller followed by an attitude controller. The first step of the control solution is the "flight mode update" which is used to offer a target mode for the UAV (e.g., "Land" and "RTL" for return to launch). Flight mode updates can be informed by a command from a radio controller when in manual mode or, as is done in this study, issued by the external onboard computer (i.e., the TX2) as part of a state machine associated with automated flight operations. Depending on the flight mode, the flight controller utilizes different control strategies. The "Land" mode is most pertinent to this study and the control logic behind precision landing is described here. The outermost control loop is the position controller that is based on a proportional-integral-derivative (PID) controller design. The position controller takes in a target vehicle position $\vec{r}_{cmd}$ and velocity $\vec{v}_{cmd}$ from the precision-landing Kalman filter (which will be illustrated in the next section) and the vehicle's actual position $\vec{r}$ and velocity $\vec{v}$ from the 24-state EKF to generate a target vehicle attitude $\vec{q}_{cmd}$ that will be fed into the attitude controller. Similarly, the attitude controller adopts a PID design for each angle axis and outputs desired angular body rates $\vec{\Omega}_{cmd}$ (i.e., $[\Omega_{cmdX}\,\Omega_{cmdY}\,\Omega_{cmdZ}]^T \in \mathbb{R}^3$ along the three axes of the vehicle's body frame) for the vehicle. At the end of the ArduCopter main loop, outputs from the attitude and position controllers are converted to absolute motor outputs (i.e., PWM values) for the specific frame type (e.g., quad, X8) and sent to the electronic speed controls (ESCs) which command each motor with a PWM output, $\vec{F}_{cmd}$ (i.e., $[F_1\,F_2...F_n]^T \in \mathbb{R}^n$, where n is the number of motors). It should be emphasized that the contributions of this work lay mainly on the TX2 side, where visual estimation of the UAV's relative position to the landing pad is provided based on computer vision methods. In contrast, the Pixhawk is used as coded with an addition of a Kalman filter for precision landing and fine tuning of the PID control parameters for precision control of the UAV.

The Nvidia Jetson TX2 companion computer running Linux (Ubuntu 16.04) constitutes the other significant part of the UAV software system. The main thread embedded on the TX2 is a finite-state machine (FSM) for automated flight operations including where and when to place wireless sensors on a structural system. The FSM is primarily focused on implementing a target search for identifying sensor locations and performing precision landing of the UAV while ensuring flight safety and efficiency. To do this, the UAV's bottom mounted camera interfaced to the TX2 will be used to search for landing locations identified with fiducial markers and to improve UAV positioning during precision landing. The DroneKit-Python API[32] is set up on the TX2 to establish communication between the Pixhawk flight controller and the TX2 using the MAVLink communication protocol. Through this low-latency communication protocol, the TX2 is able to get real-time access to the vehicle's state and to command vehicle operations.

3 | METHODOLOGY

3.1 | Sensor deployment state management

The major intellectual merit of the work is embodied in the methodology associated with automation of sensor deployment and redeployment. At the core of the work is the creation of an FSM embedded in the UAV onboard computer (i.e., TX2) that choreographs each step of the fully autonomous sensor deployment. The deployment strategy (Figure 1) is based on a structure with predetermined sensor locations defined by fiducial markers. Once sufficient data are collected, the sensor is retrieved by the UAV, transported and positioned to another installation location. This deployment strategy is executed by the UAV using the FSM embedded in the onboard TX2 computer.

The FSM approach partitions the autonomous sensor deployment method into a set of well-defined operational states with deterministic transitions between them. As shown in Figure 5, the task of deploying a wireless sensor node is split into manageable pieces such as searching for the target landing pattern in the air, hovering above the landing pattern, and performing a precision landing of the UAV for sensor placement. The FSM guides the UAV to first take off from the home position to a target height and to fly a predetermined flight path while searching for the desired fiducial marker on the structure that indicates the installation location of the wireless sensor (e.g., sensor location 1 in Figure 1). If the desired landing pattern is found, a precision landing maneuver based on computer vision is performed to land on the target pattern for placement of the sensor. After desired measurements are taken, the UAV takes off and searches for the next installation location (e.g., sensor location 2 in Figure 1). The UAV repeats this procedure until all

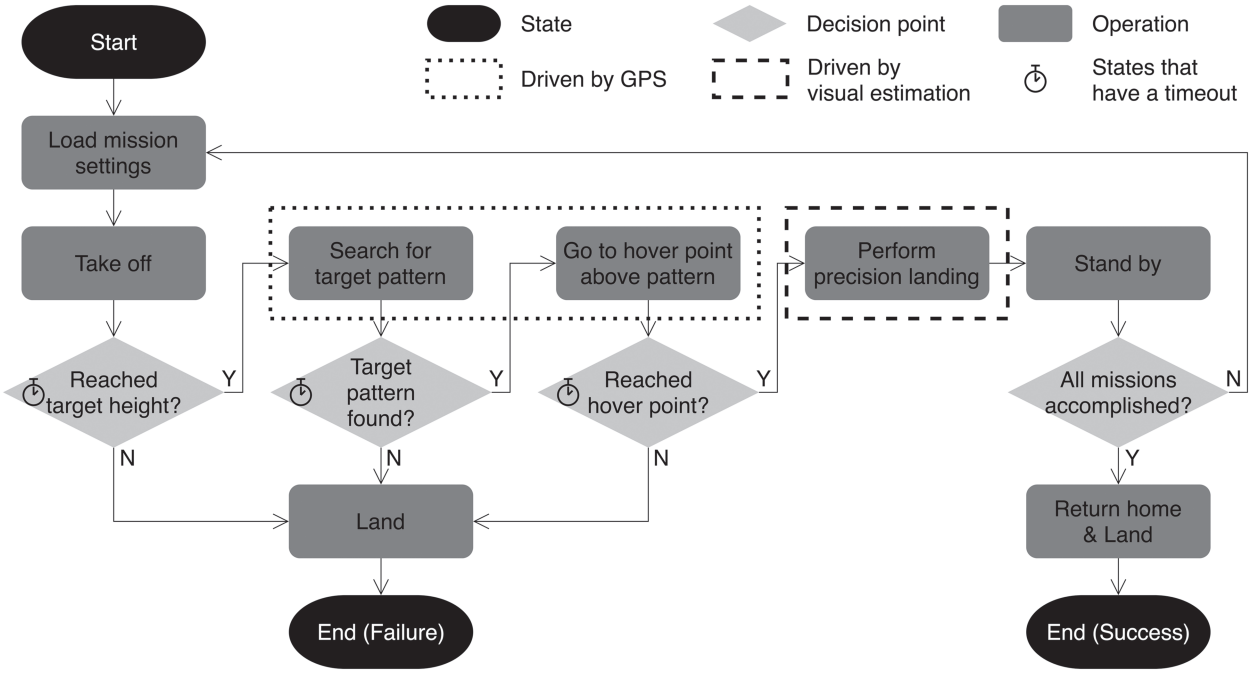


FIGURE 5 Finite-state machine for the UAV-based sensor placement mission; certain operations are only granted a limited time to stay in for battery life preservation

required locations are visited, at which point the UAV returns to its home position. A challenge with UAVs in general is their limited battery energy; this requires an efficient FSM that does not waste scarce energy. Certain states in the FSM presented in Figure 5 are only granted a limited time for the UAV to stay in so that battery life is not wasted.

In the flight operations dictated by the FSM, the UAV will rely on two primary sources of data for spatial pose estimation: vehicle state estimation data (e.g., vehicle position $\vec{r}$, velocity $\vec{v}$, and attitude $\vec{q}$) queried by the TX2 companion computer from the flight controller and camera images viewing the landing pattern which are a set of unique fiducial markers. The TX2 has a flight path defined by GPS waypoints that is communicated point by point to the flight controller over the MAVLink communication interface. The GPS data is sufficiently accurate (i.e., within meters) for guiding the UAV over large distances, but is too coarse for precision landing. Once the fiducial marker corresponding to a desired landing location is found, the UAV uses the TX2 computer to estimate with greater precision (i.e., within centimeters) the UAV position $\vec{r}$ and orientation $\vec{q}$ using camera images of the fiducial markers and a Kalman filter. Once the landing pattern is detected, the cascaded position and attitude controller (Figure 4) inside the Pixhawk controls the UAV to land using the estimated UAV pose relative to the landing pattern as an input to the control law. The processing of the camera images and relative pose estimation are done by the TX2 computer using tailored software written in Python as part of this study. The fusion of the camera estimates and other sensor data such as IMU-based measurements is implemented via a second Kalman filter (i.e., independent of the EKF) embedded in the Pixhawk flight controller by the authors.

3.2 | Landing pattern design and detection

Being able to detect the landing pattern defined by fiducial markers is fundamental to the automated sensor deployment FSM. Specific challenges include keeping the visual target within the camera's limited field of view (FoV), robust detection of the landing pattern using low-resolution images, and use of the fiducial markers for UAV state estimation for precision landing. A multiresolution tag pattern is designed to address these challenges. During a precision landing task, ground areas covered by the UAV's downward facing camera is limited by the camera's FoV and the height, dz, of the UAV. A camera from a higher altitude with a wider FoV has better coverage of the ground. More specifically, as shown in Figure 6, the maximum lateral and longitudinal ground distance covered by the camera, dx and dy, can be calculated:

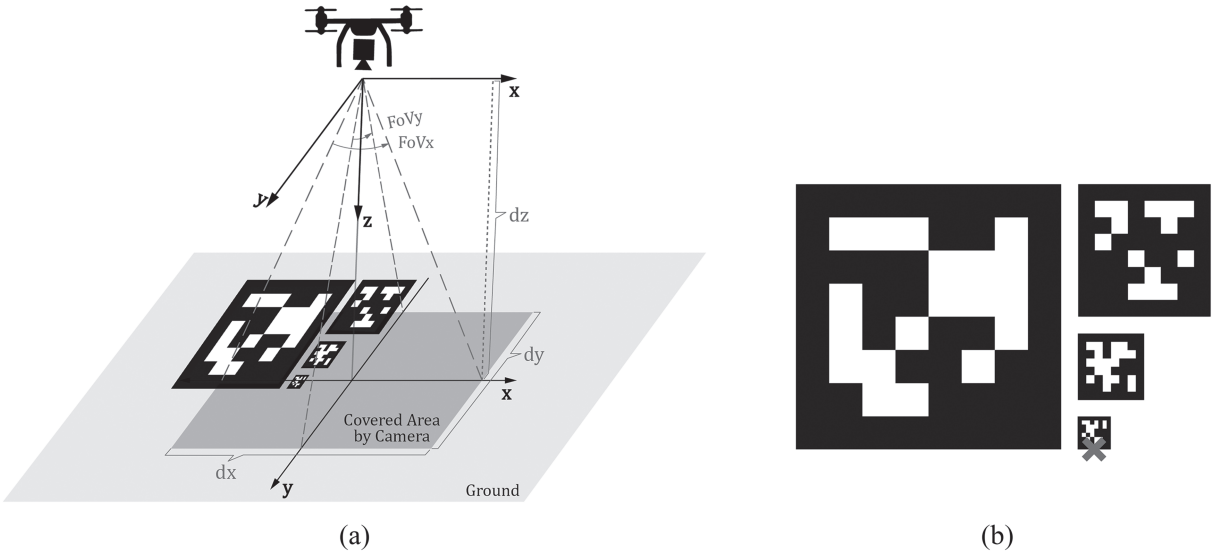


FIGURE 6 (a) Area covered on the ground by a downward facing camera restrained by FoV and height (in this case, only the two smallest tags stay completely in the camera's FoV); (b) landing pattern with four different sizes of AprilTags with the gray cross mark defining the landing spot

$$dx = 2 \cdot dz \cdot \tan\left(\frac{FoV_x}{2}\right), dy = 2 \cdot dz \cdot \tan\left(\frac{FoV_y}{2}\right), \tag{1}$$

where FoV_x and FoV_y are the camera's field of view along the x and y axis, respectively. The designed landing pattern includes fiducial markers of different sizes which are intentionally positioned as guides to the UAV at different distances from the target during landing, leading to better precision. Bigger markers allow the UAV to detect landing spots from high altitudes, but smaller markers are needed to ensure precision during landing. As the UAV descends, bigger tags gradually leave the camera's FoV while smaller ones become detectable, thereby providing a continuous navigation guide for the UAV.

The AprilTag fiducial detection system is chosen for the design of the landing pad due to its robust performance with respect to suboptimal lighting conditions, occlusion, and motion blur.[22] These black and white, QR-code-like square tags contain identification information (tag ID) and provides full pose estimation of a calibrated camera with respect to a tag. The AprilTag detection system is composed of two major components: tag detector and coding system. The detector's job is to estimate the position of potential tags in an image and the coding system enables encoding/decoding of distinguishable IDs. The detection process starts with detecting line segments by grouping together pixels with similar gradient directions and magnitude. Sequences of line segments that form a four-sided shape (i.e., possible tag boundaries) are then identified based on a recursive depth-first search method.[22] The final stage of the detection algorithm is payload decoding, where bits from the tag-relative payload field are extracted one by one. Once the data payload is determined, the coding system determines whether it is a valid tag or not. The AprilTag encoding scheme utilizes a modified lexicographic coding system that ensures minimum Hamming distance between codewords while rejecting simple patterns that commonly occur in natural scenes.[22] Different families of tags are provided by the AprilTag coding system, depending on the size of the tag (e.g., 4×4, 5×5, and 6×6 grids) and the minimum Hamming distance between codewords. In general, families with smaller grid size (and hence larger pixel size) enable detection from afar while those with a larger grid size allow larger Hamming distance thereby providing higher identification accuracy. In this study, the Tag36h11 AprilTag family (6×6 codewords with a 11 bits minimum Hamming distance) is adopted for its high detection accuracy with low processing time. A Python module is implemented in the UAV TX2 to detect the AprilTag.[33]

The strategy taken in this study is to adopt four AprilTag markers of different sizes as a multi-resolution landing pattern (Figure 6b) for precision landing. The four markers are organized in an optimal fashion to minimize jitter in the control of the UAV that can occur if the landing control algorithm switches tags for its reference during execution.[34] To further minimize jitter, this study abandons the switching of AprilTag markers during landing and uses all of the

tags identifiable to reference a single reference point which is associated with the smallest marker (since this tag is the most likely to stay in the UAV camera FoV the longest). The UAV will identify as many of the AprilTag markers it can and use all detected markers to estimate the UAV pose.

3.3 | Visual position estimation

Having detected the landing pad and identified the pixel coordinates of the detected AprilTags' feature points (i.e., tag center and corners), estimation of the position of the UAV relative to the landing point is possible. Towards this end, a robust relative position estimation method relying on both IMU and vision data is developed. To compensate for cases where visual localization data is not available, a Kalman filter is next implemented to provide continuous estimation of the UAV position relative to the landing point.

3.3.1 | Relative position estimation

Figure 7a presents the method for positioning the UAV with respect to the landing point M. In this study, a superscript on the location variable denotes the reference frame (e.g., P^A denotes the coordinates of point P with respect to frame A). Three main coordinate frames and transformations between them are illustrated. The global NED frame, denoted *Ground*, is located at the UAV's home position P (i.e., where it takes off). The North direction (x_P) is provided by the magnetometer sensor onboard the vehicle. The *UAV* body frame is set at its center of gravity (CG), point Q, with the x_Q axis pointing between two of the arms towards the front of UAV, the y_Q axis pointing to the right, and the z_Q axis pointing to the bottom of the UAV. A *Camera* frame $O(x_O, y_O, z_O)$ is defined at the optical center O. Common practice is to set x_O to right of the camera, y_O down, and z_O outwards from the camera lens. The objective is to find the relative position of the landing point M with respect to the UAV's CG, point Q, in the *Ground* frame. Because the UAV's coordinates in the *Ground* frame (i.e., Q^{Ground}) is provided as state estimations by the 24-state EKF implemented in the Pixhawk flight controller, the objective then becomes to find M's coordinates in the *Ground* frame (i.e., M^{Ground}).

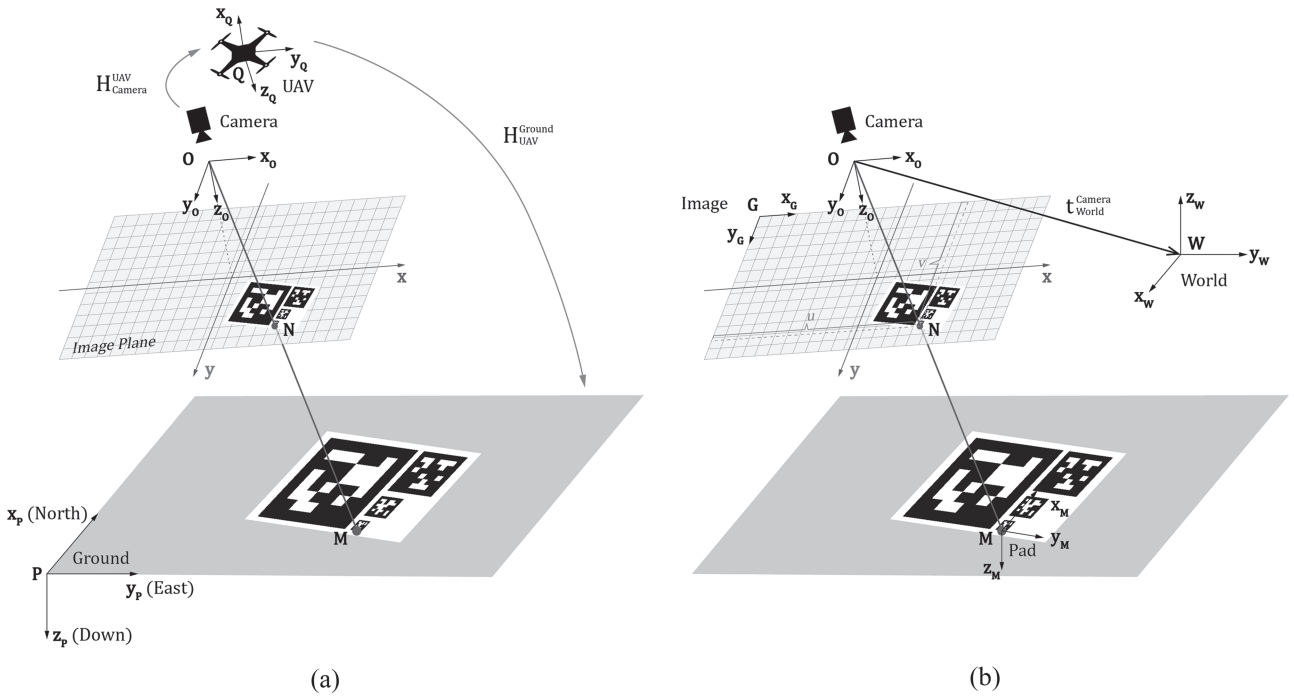


FIGURE 7 Diagram illustrating different coordinate systems and transformations between them: (a) how to compute the relative position of the UAV with respect to the landing point M in the *Ground* frame; (b) pinhole camera model showing the landing point M in the 3D *World* frame and its projection point N in the *Image* plane through the camera lens center O

The first step is to compute the landing point M's representation in the *Camera* frame, M^{Camera}. To this end, a pinhole camera model (Figure 7b) is adopted to construct the mathematical relationship between coordinates of point M in a 3D world frame and the 2D coordinates of its projection point denoted as point N in the image pixel frame. Four coordinate systems are defined in the pinhole model of Figure 7b. The *Camera* reference frame is as defined in Figure 7a. The *World* frame $W(x_W, y_W, z_W)$ is the frame of reference for absolute positioning. The *Image* coordinate frame, denoted as $G(x_G, y_G)$, is defined with the origin G at the top-left corner of the image with x_G pointing to the right. The fourth reference frame *Pad*, denoted as $M(x_M, y_M, z_M)$, is defined at the landing point M. In the *Pad* frame, y_M points to the right and z_M points perpendicular to the pad itself. Both the x_M and y_M axes are parallel to tag boundaries.

The pinhole camera model can be formulated as follows:

$$cN^{Image} = K\left[R^{Camera}_{World} | t^{Camera}_{World}\right] M^{World}, \tag{2}$$

where $c \in \mathbb{R}$ is a scaling factor, $K \in \mathbb{R}^{3\times3}$ is the camera intrinsic matrix, $\left[R^{Camera}_{World} | t^{Camera}_{World}\right] \in \mathbb{R}^{3\times4}$ is the camera extrinsic matrix representing the relative rotation and translation of frame *World* with respect to frame *Camera*, $N^{Image} = [u\, v\, 1]^T \in \mathbb{R}^{3\times1}$ and $M^{World} = [x\, y\, z\, 1]^T \in \mathbb{R}^{4\times1}$ are homogeneous coordinates of point N and M, respectively. The camera intrinsic matrix, K, is based on the camera optics and is a constant matrix that only needs to be found once. Equation 2 can be expanded as follows:

$$c\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{3}$$

Note that the translation vector t^{Camera}_{World} can be interpreted as the coordinates of the origin of the *World* frame with respect to the *Camera* frame (Figure 7b). By purposefully defining the *World* frame exactly as the *Pad* frame, the landing point M becomes the origin of the *World* frame, and M's representation with respect to the *Camera* frame is simply t^{Camera}_{World}, that is,

$$M^{Camera} = t^{Camera}_{World} \triangleq [t_1\, t_2\, t_3]^T. \tag{4}$$

M^{Camera} represents all the necessary information about the landing point that can be obtained from a single image needed for control of the UAV to land with precision on the pad. Specifically, M^{Camera}, or the translation vector t^{Camera}_{World}, defines the relative distances, Δx, Δy, and Δz, of the camera with respect to the landing pad that will be used to control the UAV landing with the control law aiming to drive t^{Camera}_{World} to zero.

The problem of estimating the pose of a calibrated camera (with known intrinsic matrix, K) based on a set of n reference 3D points and their corresponding 2D projections is commonly referred to as the Perspective-n-Point (PnP) problem. Mathematically, the PnP problem can be defined as given a set of n 3D world coordinates-2D image coordinates pairs, determine the camera extrinsic matrix $\left[R^{Camera}_{World} | t^{Camera}_{World}\right]$ (Equation 2). Existing solutions to the PnP problem can be classified into two methods: optimization-based iterative methods (solved by minimizing a properly defined cost function)[35] and closed form methods (solving the equation directly).[36] Making use of the open-source computer vision library, OpenCV,[37] an iterative method called solvePnP[38] is selected here. The cost function used by solvePnP is the reprojection error, which is defined as the sum of squared distances between the observed projection points and those calculated in each iteration. Solutions of the extrinsic matrix $\left[R^{Camera}_{World} | t^{Camera}_{World}\right]$ that minimize the reprojection error are found based on the well-known Levenberg–Marquardt algorithm.[39,40] The translation vector t^{Camera}_{World} (i.e., t_1, t_2, and t_3) is then the relative location of the camera to the pad in the *World* frame; it is this vector that is used by the subsequent Kalman filter used to estimate the UAV position during controlled landing.

The compute-intensive solvePnP algorithm is implemented in the TX2. After obtaining the translation vector t^{Camera}_{World} from each image, MAVLink messages encoding t_1, t_2, and t_3 are sent from the TX2 to the flight controller (Figure 4). These visual estimations are used as a measurement update for a precision-landing Kalman filter that will be illustrated next.

Once M^{Camera} is computed, a set of homogeneous transformations can be applied to get M^{Ground} based on rigid motions between different frames:

$$M^{Ground} = H^{Ground}_{UAV} H^{UAV}_{Camera} M^{Camera}, \tag{5}$$

where M^{Ground} and M^{Camera} are 4×1 homogeneous coordinates augmenting the original 3×1 coordinates by a fourth component of 1. A homogeneous transformation matrix H^A_B (4×4) is nothing but a compact way to include both the relative rotation R^A_B and translation t^A_B between two frames A and B, that is,

$$H^A_B = \begin{bmatrix} R^A_B & t^A_B \\ 0 & 1 \end{bmatrix}. \tag{6}$$

H^{Ground}_{UAV} comes from the attitude and position estimations for the UAV in real time, and H^{UAV}_{Camera} is a preset constant dependent on how and where the camera is mounted on the UAV.

As a side note, the rotation matrix R^{Camera}_{World}, or specifically R^{Camera}_{Pad}, produced by the solvePnP algorithm is utilized to line up the *Camera* frame and the *Pad* frame such that the camera's x_O axis coincides with the pad's y_M axis. Before precision landing starts, the vehicle is commanded to yaw an appropriate angle based on this rotation matrix. In this way, the orientation of the UAV is deterministic with respect to the landing pad when it lands.

3.3.2 | Kalman filter for landing position estimation

The IMU can be further exploited to increase the robustness and accuracy of the vision-based relative position estimation. Visual and inertial fusion has been an active topic of research to address accurate and reliable localization and mapping in a wide range of robotic oriented applications. Existing approaches found in the literature can be broadly classified into two categories: batch nonlinear optimization methods[41–43] and recursive filtering methods.[44,45] The optimization methods jointly minimize errors from both the IMU and vision measurements, while filter based methods commonly use IMU measurements for state propagation with updates originating from visual observations. Nonlinear optimization methods are higher performing, but their increased accuracy comes at a cost of more computational resources. Hence, a recursive linear Kalman filter is adopted in this work due to its simplicity and the flight controller's limited computing power.

A standard visual-inertial filtering method requires a state vector involving states of both the vehicle and the landing pad (e.g., their positions and velocities). When visual data is not available, state estimation for the UAV is provided by the 24-state EKF implemented in the Pixhawk flight controller stack. When an image is available, the original 24-dimension state vector would be augmented to include the landing pad's states. This approach would require a large number of modifications to the original flight controller EKF codebase shown in Figure 4. Alternatively, a less accurate but more efficient approach taken herein is adding a second Kalman filter for relative position and velocity estimations of the landing pad while leaving the state estimates for the UAV from the original 24-state EKF filter unchanged. State estimations from the 24-state EKF can be used as inputs to the second Kalman filter, while visual estimation results from the previous section will be used as measurement updates (Figure 4).

The second Kalman filter is established as follows. As the dynamics of the UAV are loosely coupled in the x_P, y_P, and z_P directions in the *Ground* frame, they are modeled independently during the precision landing process. In the z_P direction, a standard landing maneuver is commanded. On the horizontal plane, two discrete Kalman filters are used independently: one in the x_P (North) direction and another in the y_P (East) direction. Without loss of generality, the Kalman filter in the x_P direction is illustrated. The states of the Kalman filter are the relative position and relative velocity of the landing pad with respect to the UAV, $s = \left[x^{rel}\, v^{rel}\right]^T$. The next state s_{k+1} is propagated from the currents state s_k using the following motion model:

$$s_{k+1} = \begin{bmatrix} x^{rel}_{k+1} \\ v^{rel}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x^{rel}_k \\ v^{rel}_k \end{bmatrix} + \begin{bmatrix} 0 \\ \delta v^{rel}_k \end{bmatrix} + \begin{bmatrix} 0 \\ \delta v_{noise}{}^{rel}_k \end{bmatrix} \triangleq F_k s_k + G_k u_k + q_k. \tag{7}$$

The Kalman filter assumes a constant relative speed in δt, which is reasonable because the filter is updated at 400 Hz. The controlled input $u = \left[0\, \delta v^{rel}\right]^T$ is the negative of the UAV's velocity change over the timestep δt. The process noise $q_k = \left[0\, \delta v_{noise_k^{rel}}\right]^T \tilde{N}(0, Q_k)$ is set to be the estimated accelerometer noise times δt. Both δv^{rel} and $\delta v_{noise}{}^{rel}$ comes directly from the original 24-state EKF. Relative distance is measured and updated when visual data arrives. The measurement model is simply:

$$m_k = x_{meas_k^{rel}} = [1\ 0] \begin{bmatrix} x_k^{rel} \\ v_k^{rel} \end{bmatrix} + \delta m_{noise_k^{rel}} \triangleq H_k s_k + r_k, \tag{8}$$

where z_k comes from the relative position estimates output from execution of solvePnP on the TX2 using the image data. The measurement noise $r_k = \delta m_{noise_k^{rel}} \tilde{N}(0, R_k)$ is assigned to be 2% of the UAV's distance to landing point, which works well in practice. Given the linear state space model of the relative position Equations 7 and 8, a Kalman filter is developed to predict the state, s_{k+1}. The Kalman filter predicts the state mean, $\widehat{s}_{k+1}$, and the state covariance matrix, $\widehat{P}_{k+1}$, before updating the gain of the Kalman filter, K_{k+1}, state, s_{k+1}, and state covariance matrix, P_{k+1}.

The discrete Kalman filter is run on the Pixhawk flight controller since the real-time operating system provides a precise time base. The use of a Kalman filter allows compensation for bad visual localization data or even the loss of it. Erroneous and inaccurate state estimates are detected based on the normalized innovations squared (NIS) metric.[46] In cases when a visual update is unavailable (e.g., if the camera fails to detect the landing point), the filter will only make a prediction of the current iteration and compensates in the next update step when the image is available. This allows the system to recover from several dropped camera frames.

3.3.3 | Visual-inertial synchronization

Conducting image capture and processing as close to real-time as possible is critical for a healthy Kalman filter. High latency could result in incorrect position estimations. A seemingly simple frame polling command from the TX2 to the camera introduces latency because polling involves image processing in the camera, USB transfer of the image, and image decoding by the TX2. To reduce latency present in the polling process, an image capture background thread is adopted by the TX2 that continually polls the camera and labels each resulting frame with a timestamp. Image processing is carried out in the main thread with landing point estimations transferred to the flight controller via the MAVLink protocol when available. Because of the latency present in the camera image pipeline, an observation of the landing pad is received by the filter framework with a delay. To synchronize inertial and visual data, a buffer of IMU measurements is maintained since the IMU data is updated faster (400 Hz) than the image measurements (about 30 Hz). Once the Pixhawk receives a visual position estimate, it performs an update step of the Kalman filter on the delayed time horizon, and then predicts forward to the current time using the buffered IMU data. Figure 8 illustrates the data fusion process.

4 | EXPERIMENTS AND RESULTS

Two different sets of experiments were designed to quantitatively assess the proposed UAV-based vision system for precision placement of wireless sensors on structures. The first round of experiments (Section 4.1) focused on testing each individual system component such as position estimation of the UAV relative to the landing pattern and the control method used for precision landing of the UAV. These experiments were conducted using the 3DR X8 UAV mainly due to its sturdy frame design (allowing for possible crashes during experimental validation) and relatively longer flight times (15 min with a 1 kg payload). The second round of experiments (Section 4.2) were intended to validate the concept of deploying sensors on a structure in a modular fashion. Two QAV210 UAVs carrying cameras and wireless sensors with accelerometers were used to perform modal analysis of a beam structure. The case study featured fully autonomous operations of the two UAVs carrying a Martlet wireless sensor node that is programmed to safely land on a simply supported beam with a restricted surface area. The two QAV210s are programmed to move the sensors so as to accurately identify the beam mode shape.

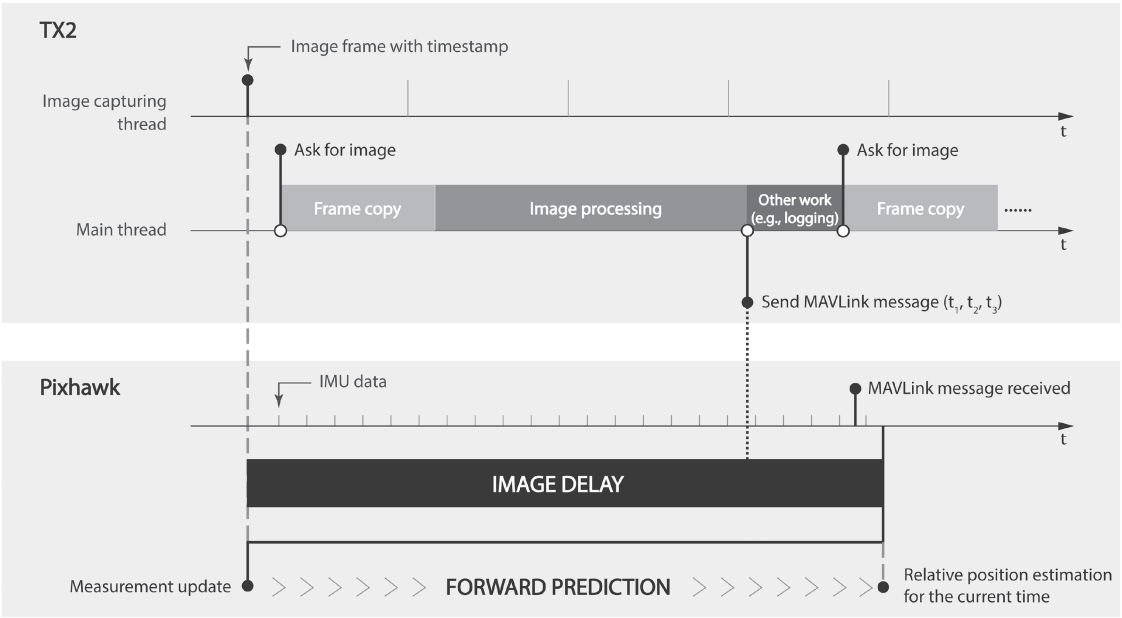


FIGURE 8 Visual-inertial synchronization: the total image delay is the lapse of time from the camera shutter time to the moment the Pixhawk executes a measurement update. Kalman filter is run on a delayed time horizon on the Pixhawk. Results from the Kalman filter are predicted forward to produce state estimations for the current time utilizing an IMU buffer

Field experiments were performed in M-Air, a netted outdoor flying lab ($22.4 \times 36.6 \times 15.2$ m^3) designed for UAV research located on the campus of the University of Michigan. During testing, wind conditions were considered mild (with wind speed under 5.4 m/s but with occasional gusts up to 6.7 m/s) throughout the duration of the experiments. A Qualisys motion capture system is permanently integrated into M-Air and includes 30 cameras installed around the facility to provide accurate (mm-level) tracking of object motion. Retroreflective passive markers were mounted on the objects of interest, in this case, this study's 3DR X8 UAV and its landing pad (Figure 2a), for tracking purposes during the first experiment focused on tracking the landing process of the 3DR X8. The cameras' threshold was adjusted so that only the bright reflective markers were captured. UAV and landing pad positioning data were reported by the Qualisys IR system in real-time at 60 Hz and were used as ground truth for the validation studies. It should be noted that the Qualisys IR system was not involved in the second experiment as the two QAV210 UAVs used only onboard vision and computing for autonomous missions.

4.1 | UAV system component testing

For testing and validation purposes, the landing pattern was designed with four AprilTags (Figure 6b) with side lengths of 22.4, 11.2, 5.6, and 2.8 cm. The largest AprilTag can be reliably detected from as high as 12 m from the air. The smallest ApilTag fits into the X8 camera's FoV even when the UAV sits on the ground over the landing spot (there is a 9.2-cm distance between the camera and the ground).

4.1.1 | SolvePnP validation

First, the performance levels of relative position estimation method proposed in Section 3.3.1 were evaluated. Two types of test flights were performed to quantify the performance of the solvePnP algorithm for estimating the relative position of the landing point, M, with respect to the camera lens of the UAV (i.e., M^{Camera}). In the first round of test flights, the UAV was kept in a relatively close position above the AprilTag landing pattern with the UAV continuously estimating its position relative to the landing point, M. The Qualisys IR motion capture system was used to determine the relative position of the UAV with respect the landing pattern as ground truth. The second round of testing centered on how

distance affects the estimation accuracy of the embedded algorithm. In these tests, the UAV took off from the landing pad and rose up to about 12 m.

Test results for two of the flights are shown in Figure 9. Estimations of M^{Camera} (t_1, t_2, and t_3 from solvePnP) are shown in red plus signs and ground truth measurements from the motion capture system are shown in solid blue lines. Figure 9a–c shows results of a flight where the UAV was flown relatively close to the landing pattern at a height of roughly about 1.4 m, while Figure 9d–f shows another flight where the UAV was slowly flown away from the landing pattern getting to a height of more than 12 m. In both test flights, the solvePnP algorithm is able to provide estimations that follow the ground truth tightly in all three directions, which demonstrates the accuracy of the algorithm. Root mean square error (RMSE) for the estimations are shown in Table 1. For the first test flight, RMSEs in all three directions are well under 3 cm, which indicates the ability for the UAV to precisely land.

The second test flight shows that the landing pattern can be detected by the camera from as far away as 12 m. However, as the distance between the camera and landing point (i.e., $\left\|\overrightarrow{OM}\right\|$) grows larger, the estimation accuracy decreases. The accuracy of the position estimation algorithm was assessed as a function of the relative distance $\left\|\overrightarrow{OM}\right\|$. Figure 10 presents the estimation errors with respect to the relative distance in all three directions. Also shown in the figure is the number of AprilTags detected and used to compute the relative distances indicated by blue stars. As shown, only one AprilTag is detected when the vehicle is far from the pattern ($\left\|\overrightarrow{OM}\right\| > 6.0$ m) and close to the pattern

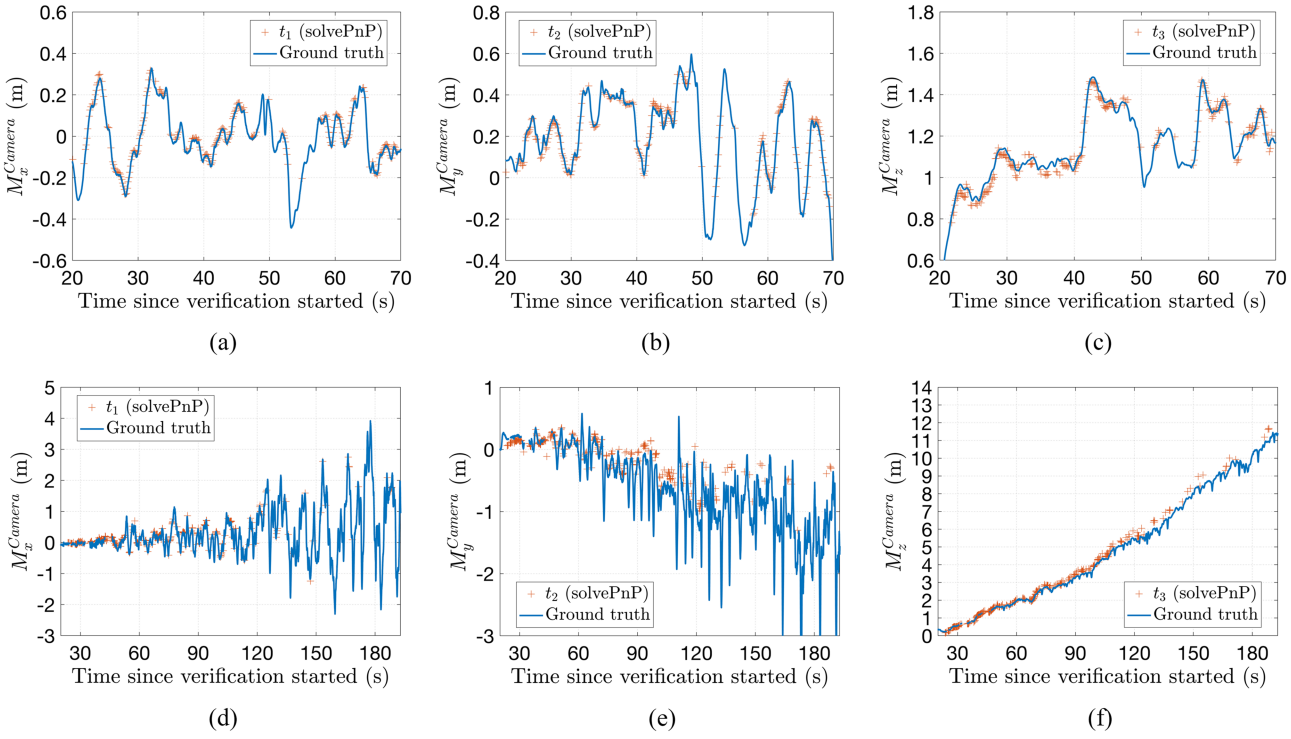


FIGURE 9 Estimation of M^{Camera}: (a–c) UAV flown relatively close to landing pattern at a distance roughly about 1.4 m; (d–f) UAV took off from the landing point and slowly flown away

TABLE 1 RMSEs of the solvePnP position estimation algorithm (units are in meters)

	M_x^{Camera}	M_y^{Camera}	M_z^{Camera}
1st flight (Figure 12a–c)	0.0107	0.0165	0.0269
2nd flight (Figure 12d–f)	0.0637	0.1282	0.2008

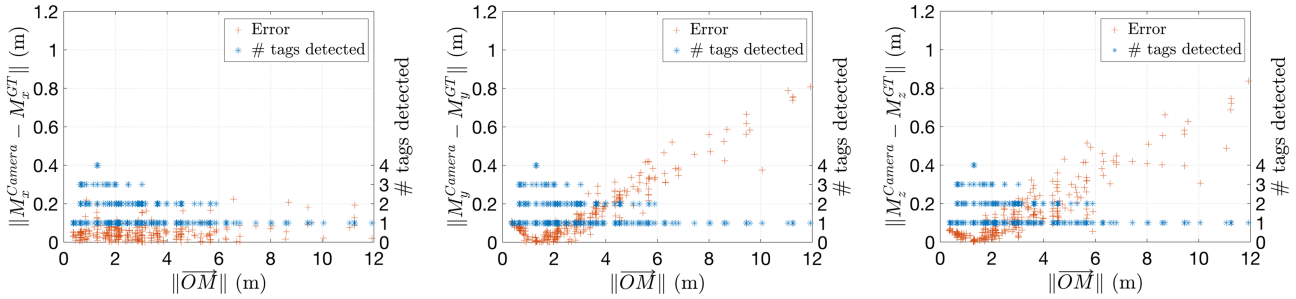


FIGURE 10 Estimation errors with respect to the relative distance between the UAV and the landing pattern. Overlayed are the number of AprilTags detected and used to compute relative distances

($\left\|\overrightarrow{OM}\right\| < 0.6$ m), which proves the necessity of the inclusion of all four AprilTags in the landing pattern. Figure 10b,c shows a clear correlation between the number of detected tags and the estimation accuracy. When all four AprilTags in the pattern are detected ($\left\|\overrightarrow{OM}\right\| \approx 1.6$ m), the estimation error is the smallest (close to zero).

Another interesting finding is that comparing Figure 10a,b, when $\left\|\overrightarrow{OM}\right\|$ is relatively large, the estimation errors in the *Camera*'s x_O direction are smaller than those in the y_O direction. This is possibly due to the pattern's larger overall length in the x_O direction (thus more accurate pixel coordinates of the feature points and better distance estimation).

4.1.2 | Image data transmission delays

It is important for attitude data from the Pixhawk flight controller and the camera-based estimation of the UAV position relative the landing pattern to be synchronized. The UAV was flown over a landing pattern and the time delay between image capture (i.e., opening of the camera shutter) and transmission of extracted UAV position information (i.e., t_1, t_2, and t_3) from the TX2 to the Pixhawk over the MAVLink interface was calculated. This delay is shown in Figure 11 as the solid red curve with cross markers. The delay, which is 31.5 ms on average, is mostly due to the time needed to load the image from the camera and process the image. The processing time for the TX2 to process the image of the landing pattern is shown as the green curve with plus markers in Figure 11. The computation time is on average 13.9 ms and less variable. Also, this computation time is not affected by the number of AprilTags (indicated by blue star markers in Figure 11) used in calculation of the UAV relative position. There is a high level of variability in the total delay (red curve with cross markers) not seen in the computation time (green curve with plus markers) of the UAV position estimation. This variability is associated with stochastic delays of the TX2 operating system (which is not real-time) when servicing the MAVLink interface and executing image capturing in the background. Once the relative position estimate is determined, the TX2 will transmit its relative position to the Pixhawk controller. The MAVLink interface operates at a baud rate of 921,600 and requires about 17.8 ms to transmit its data. If average total delay of the TX2 getting an image from the camera, calculating the relative UAV position using the image, and sending out the relative position is 31.5 $\pm$ 11.1 ms, then the total delay for the Pixhawk controller is 49.3 $\pm$ 11.1 ms. With the Pixhawk generating attitude data at 400 Hz and an average delay of image data at 49.3 ms, then an IMU buffer (discussed in Section 3.3.3) is programmed to have 20 or more data points.

4.1.3 | Precision landing

Next, the UAV's capability of precision landing was tested thoroughly. Figure 12 shows a sequence of snapshots from the onboard webcam during a typical precision landing. Detected AprilTags are highlighted in each image with the corners and centers of each detected tag marked by orange dots. Based on these feature points, pixel coordinates for the fixed landing point M are computed and marked with a red cross. M^{Camera} is further extracted using the direct method

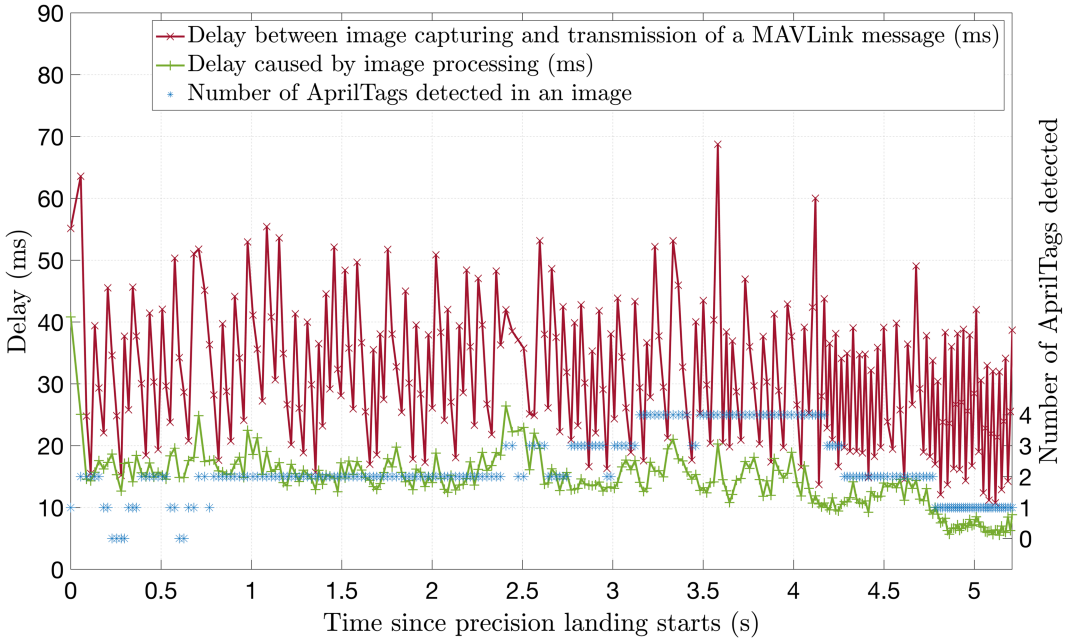


FIGURE 11 Image data transmission delay: the total delay equals the delay from image capturing to the transmission of a MAVLink message by the TX2 (red curve, on average 31.5 ms) plus the actual transmission time of the message from the TX2 to the Pixhawk (about 17.8 ms, not shown in the figure). IMU buffer size is set to 20 to compensate for the delayed visual measurement

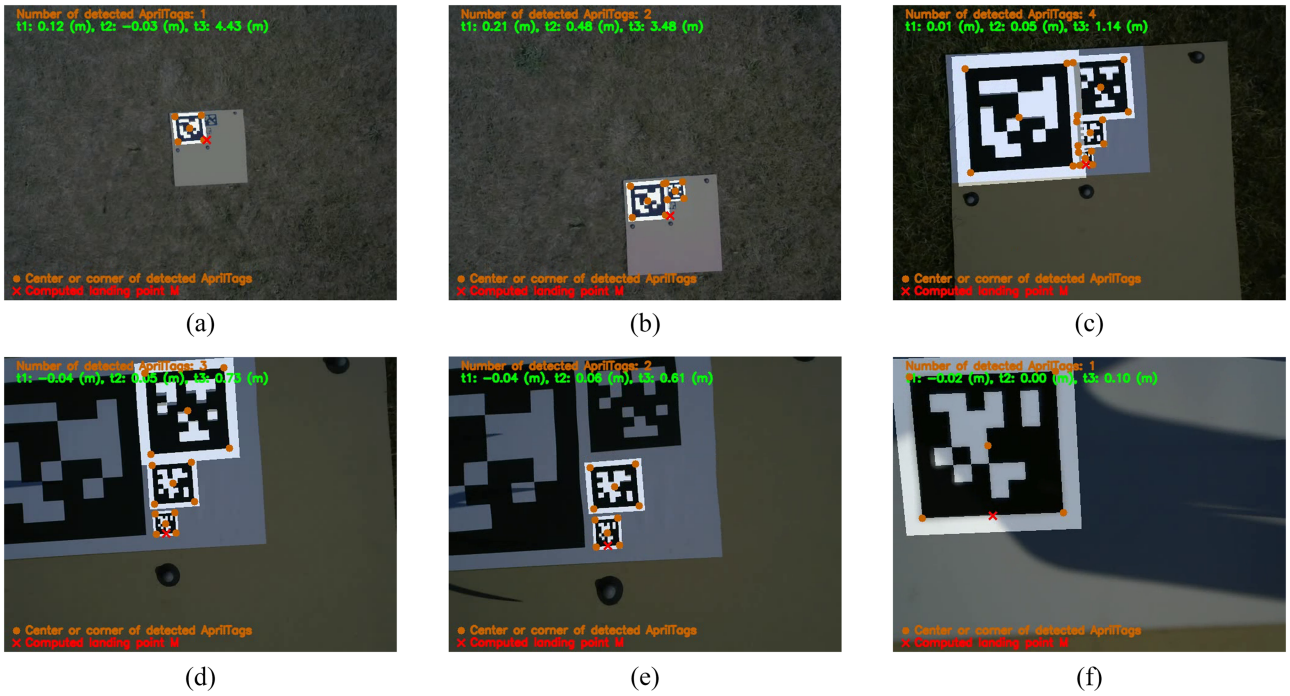


FIGURE 12 A sequence of images captured by the webcam during precision landing (computed landing point M and visual estimations from the direct method are presented in each image), roughly: (a) 4.43 m; (b) 3.46 m; (c): 1.14 m; (d) 0.73 m; (e) 0.61 m; (f) 0.10 m height

and shown on top of each frame (t_1, t_2, t_3 in green). As shown in the sequence of snapshots, only the largest AprilTag among the four is detectable at the very beginning of the landing process when the UAV is at about 4.43 m above the pattern (Figure 12a). Smaller AprilTags gradually come into the camera's FoV as the UAV descends. At the height of about 1.14 m, all four AprilTags are successfully detected (Figure 12c). After that, larger AprilTags slowly leave the camera's FoV and only the smaller AprilTags contribute to visual estimations and provide guidance to the landing vehicle.

Figure 12f shows the camera view when the vehicle lands. Relative horizontal distance from the camera lens to the desired landing point at this final stage is only 0.02 m.

Figure 13 illustrates state estimation results from the discrete Kalman filter during one typical landing. The Kalman filter shows good performance along the whole landing trajectory. Relative distance estimations (red curves) in both the Ground frame's x_P (North) and y_P (East) directions follow the ground truth (blue curves) well. Green vertical lines indicate the moments when a visual measurement is received by the Pixhawk and used to update the Kalman filter. When visual measurements are not available (e.g., during 21–23 s in Figure 13a,b, the landing pattern leaves the camera's FOV due to the UAV's aggressive maneuvers to correct its position), the Kalman filter is still able to provide estimations by only executing the prediction step. A set of landings are performed 25 times to assess the repeatability of the landing and to quantify landing precision. Figure 14 illustrates these 25 landing trajectories and their associated landing locations with respect to the desired landing point. The trajectory data shown in Figure 14a are recorded by the motion capture system. Below a height of 2.5 m, all of the landing trajectories are within a deviation of 25 cm from the desired landing point in the horizontal plane. All landing maneuvers land the vehicle within 15 cm from the desired landing point with 22 in a circle with a radius of 10 cm (Figure 14b).

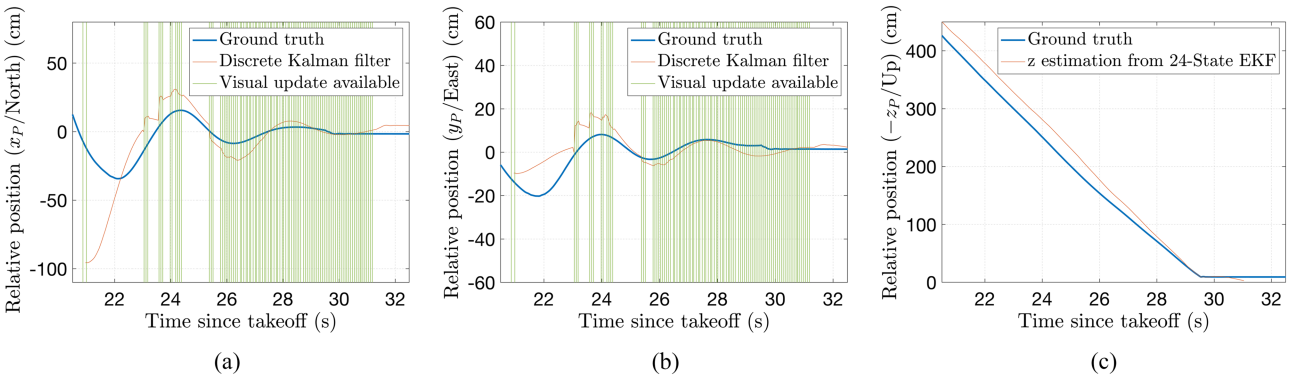


FIGURE 13 Twenty-five landing trajectories and associated landing spots: (a) trajectories captured by Qualisys IR system in red with blue circles indicating a 25-cm deviation from the desired landing point in the horizontal plane; (b) landing accuracy with landing positions indicated by red crosses

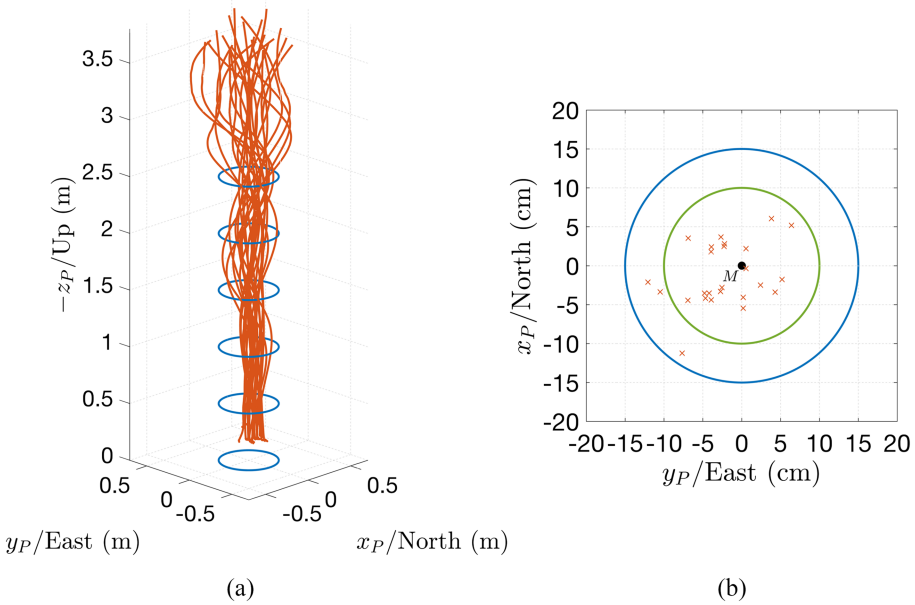


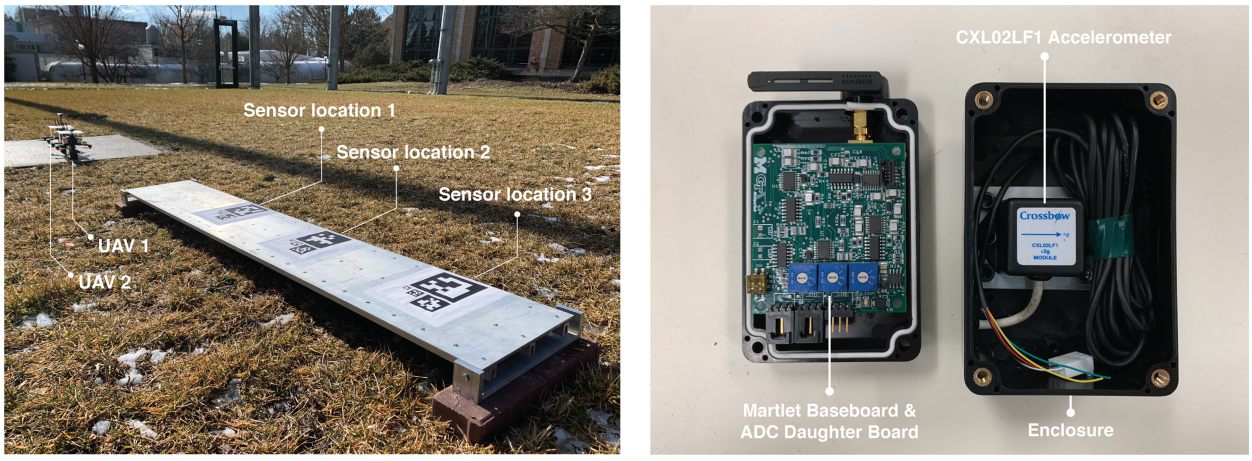
FIGURE 14 Relative distance estimations in the *Ground* coordinate system during a typical precision landing: (a) x_P (north); (b) y_P (east); (c) $-z_P$ (up)

4.2 | Autonomous modal analysis of a structural beam

The objectives of the second case study are to evaluate the feasibility of UAVs to: (1) autonomously place wireless sensor nodes on a simply supported beam structure; (2) localize the sensors on the structure; (3) collect ambient acceleration data from the structure; and (4) perform modal analysis of the beam. Towards this end, experiments were performed in M-Air using a simply support aluminum beam (182.5 cm long, 30.5 cm wide, and 0.6 cm thick). Three different sets of landing patterns are attached on the beam along its longitudinal length representing target sensor locations equidistant from one another. Two QAV210 UAVs were adopted to each carry a sensor enclosure containing a Martlet sensing node and an accelerometer sensor. The sensor enclosure is firmly mounted on the bottom of the UAV and moves from location to location on the beam as the UAV takes off and lands. When landed, the UAV's self-weight (around 1.1 kg) ensures a firm connection between the sensor and the beam. Attached on top of each UAV is another AprilTag used for localization of the UAV-sensor pair on the beam using tailored visual algorithms. Figure 15 shows the setup of the case study.

The experiment is performed in several steps. First, *UAV 1* takes off, searches in the air for the landing pattern of *sensor location 1*, and delivers the sensing node onto the beam by landing on the pattern. Once *UAV 1* lands, *UAV 2* takes off, flies above the beam and *UAV 1*, and delivers its wireless sensing node to *sensor location 2*. While *UAV 2* is in the air, it is also able to register the position of the landed *UAV 1* using camera data of the AprilTag on top of *UAV 1*. This allows the precise location of *UAV 1* on the beam to be determined. Soon after the landing of *UAV 2*, a human operator strikes the beam with a modal hammer while both UAVs command their wireless sensing nodes to collect and transmit acceleration data to a ground PC. The hammer impulse is equivalent to white noise ambient excitation expected in a real, operational structure. *UAV 1* then takes off again and moves from *sensor location 1* to *sensor location 3*. Position registration of *UAV 2* is accomplished during this process using images of its overhead AprilTag captured by *UAV 1*. Upon landing of *UAV 1*, the human operator excites the beam again and data collected by the accelerometers now at different locations on the beam are transmitted back to the ground PC. Lastly *UAV 2* is commanded to fly over *UAV 1* and leave the beam in order to register *UAV 1*'s last position. At the final stage of the experiment, modal analysis of the beam is performed using the acceleration response data collected at the three locations along the beam length. The frequency domain decomposition (FDD)[47] method is used to assemble two-point mode shapes for each sensor configuration; overlap in the mode shapes allow them to be stitched together to from global modes of the beam structure. The experiment is fully autonomous with the human operator only intervening to impulse the beam structure which would not be necessary in applications in real civil engineering structures with ambient vibrations.

As shown in Figure 16, the two QAV210s successfully positioned the Martlet wireless sensing nodes on the beam after carefully following each operation in the FSM. Position registration of the UAV is accomplished by detecting both



(a) (b)

FIGURE 15 (a) two QAV210s sit on the ground, ready to deliver wireless sensors to measure beam accelerations (landing patterns are attached to positions where acceleration measurements are desired); (b) the wireless sensing node attached on the bottom of the QAV210 (left: the Martlet baseboard together with the ADC daughter board; right: the Crossbow accelerometer)

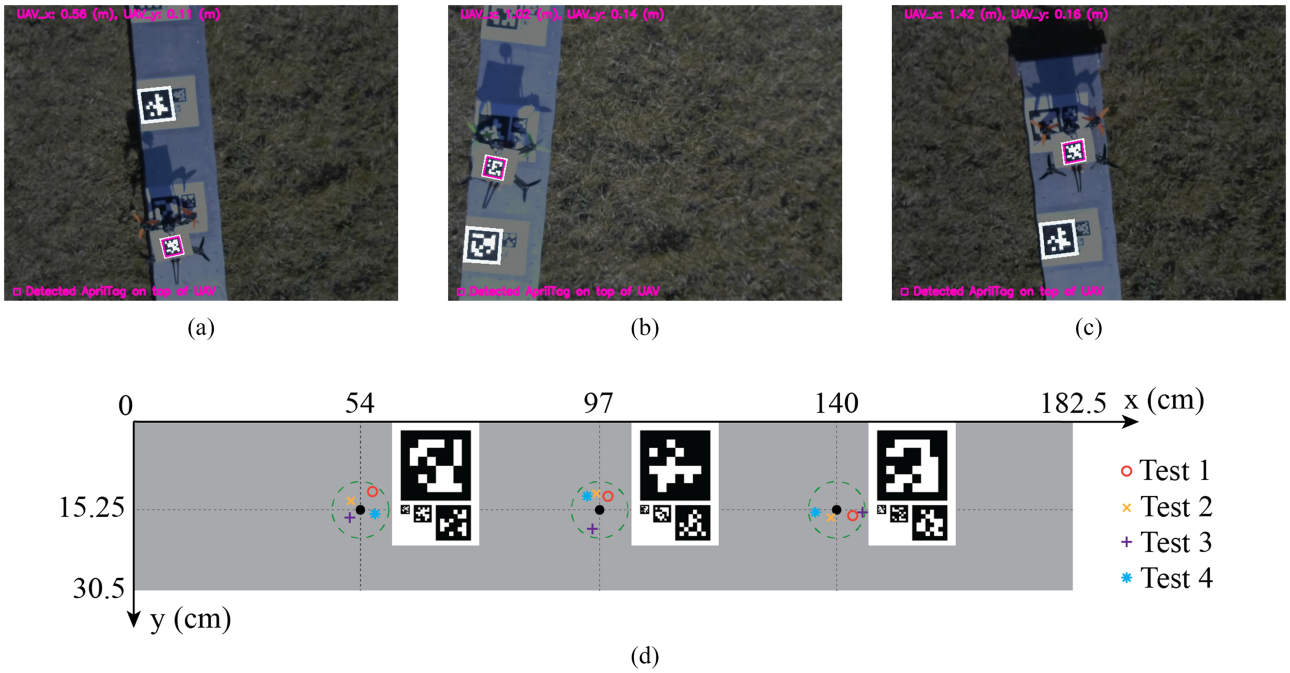


FIGURE 16 Spatial registration of wireless sensors on the beam: (a–c) position of the QAV210 landed on the beam; (d) locations of UAVs over four repeated tests (locations manually measured)

its overhead AprilTag and at least one more AprilTag on the beam at the same time. Computed UAV positions are shown in the top-left corner of the snapshots. Estimation error is within 2 cm when compared to ground truth of the UAV positions using a measuring tape. Figure 16d illustrates the measured locations of the landed UAVs (based on manual measurement with a ruler) after repeating the test four times. All landing locations are within 5 cm (shown with dashed circle with 5-cm radius) from their desired locations.

After the human operator strikes the beam with a modal hammer, the wireless sensing nodes collect acceleration data at 100 Hz as shown in Figure 17a for the first test. The FDD method is then used to extract the natural frequencies and mode shapes of the beam from the collected time history data. Figure 17b shows the singular values of the power spectral density (PSD) function matrix for the first and second sensor locations of the first test. As it appears, the first and second natural frequencies of the beam are at 13 and 43 Hz. The theoretical natural frequencies and mode shapes of a simply supported Euler–Bernoulli beam can be computed using the beam's properties ($E = 6.89 \times 10^{10}$ N/m^2, $I = 4.6785 \times 10^{-8}$ m^4, $\rho = 2.7 \times 10^3$ kg/m^3, $A = 2.31 \times 10^{-3}$ m^2, $L = 1.825$ m). The theoretical 1st and 2nd natural frequencies are determined to be 10.7 and 42.6 Hz, which are in strong agreement to the experimental results. To get the complete mode shape of beam, mode shapes (Figure 17c,d) generated from the two strikes are stitched together using the common point sensor location 2. A detailed comparison of the experimental and theoretical mode shape values is provided in Table 2. It should be noted that the mode shape values plotted in Figure 17. correspond to the actual measurement location as measured manually (as can be seen by the mode shape values for all tests not falling on a consistent vertical line). For the four different executions of the experiment, there is strong agreement between the experimental and theoretical mode shapes. Whereas the field tests were successful, and the results are reasonable, future work is needed to further reduce the modal errors such as improving the UAV controller and landing pattern design for a better landing accuracy and ensuring a firmer connection between the sensor and the structure.

4.3 | Discussion and limitations

While this research successfully demonstrates proof-of-concept trails of UAVs with structural sensors integrated precisely landing on structures using fiducial markers, there are remaining challenges inherent to scaling this approach to large and complex operational structures. The installation of fiducial markers (such as AprilTag patterns) on structures

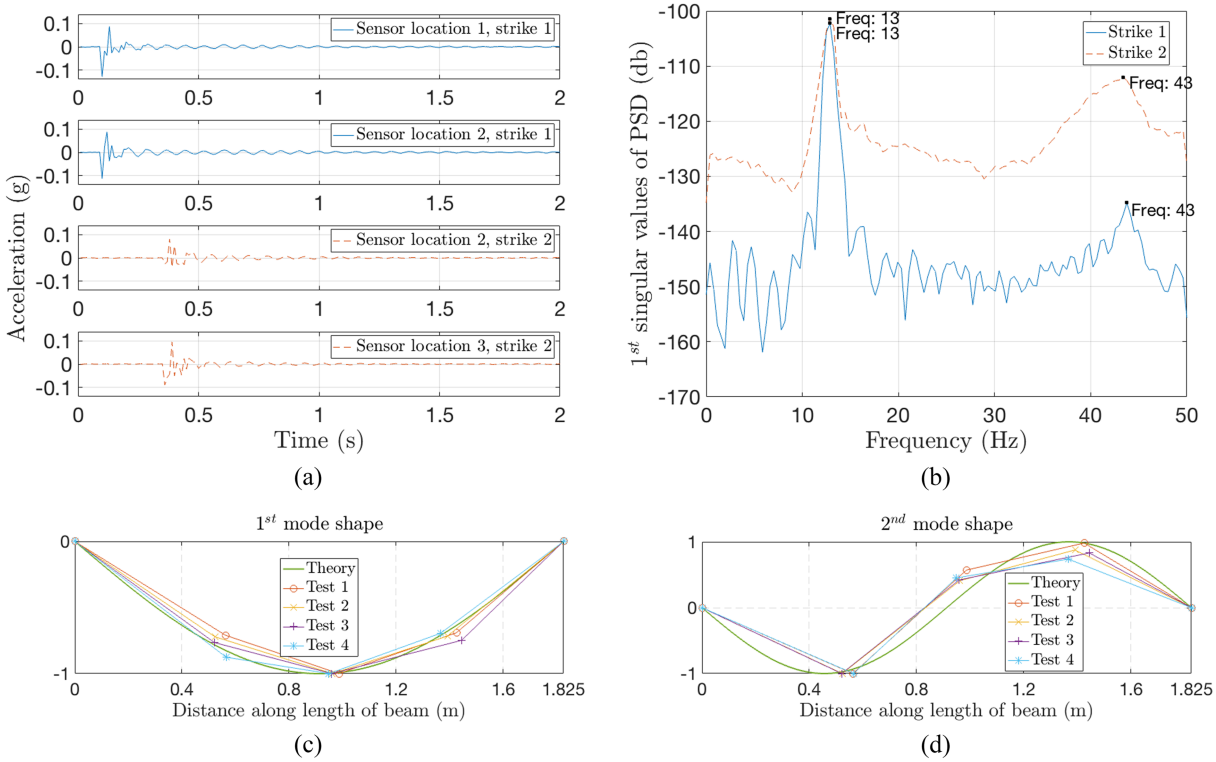


FIGURE 17 Modal analysis of the simply supported beam: (a) raw acceleration data from Martlet sensing nodes; (b) singular values of the PSD matrices; (c,d) first and second mode shape of the beam

TABLE 2 Comparison of mode shapes between experimental and theoretical values

		1st mode shape			2nd mode shape		
		Experiment	**Theory**	**Error**	**Experiment**	**Theory**	**Error**
Sensor loc. 1	Test 1	−0.7102	−0.8244	0.1142	−1.0000	−0.9332	0.0668
	Test 2	−0.7213	−0.7835	0.0622	−1.0000	−0.9737	0.0263
	Test 3	−0.7649	−0.7814	0.0165	−1.0000	−0.9753	0.0247
	Test 4	−0.8759	−0.8264	0.0495	−1.0000	−0.9307	0.0693
Sensor loc. 2	Test 1	−1.0000	−0.9920	0.0080	0.5708	0.2504	0.3204
	Test 2	−1.0000	−0.9964	0.0036	0.4103	0.1696	0.2407
	Test 3	−1.0000	−0.9968	0.0032	0.4210	0.1594	0.2616
	Test 4	−1.0000	−0.9980	0.0020	0.4520	0.1253	0.3267
Sensor loc. 3	Test 1	−0.6908	−0.6354	0.0544	0.9812	0.9813	0.0001
	Test 2	−0.7170	−0.6795	0.0375	0.8765	0.9971	0.1206
	Test 3	−0.7486	−0.6098	0.1388	0.8314	0.9666	0.1352
	Test 4	−0.6947	−0.7104	0.0157	0.7379	0.9998	0.2619

required for precise localization of the UAVs can be challenging to deploy. Human-based installation in existing structures seems inevitable in the short-term but this may come with some inherent cost and require safe access to key areas of the structure for the installer. Safe access could be challenging for a number of structures such as those in remote areas and ones with complex and large sizes (e.g., skyrises and long-space bridges). Incorporating fiducial markers into the structure during construction is another viable solution but requires planning beforehand. In this research,

AprilTags are all placed horizontally on surface with the UAV landing on the landing pattern. However, the usage of markers is not limited to a horizontal orientation. AprilTags can be put on vertical or inclined surfaces as long as they are within the camera's FoV and their location and pose relative to a predefined flat landing position on the structure is known a priori. Also, the UAV need not have to land on the pattern itself. In this work, a set of standard square tags with varied sizes is adopted to accommodate detection from a wide range of distances resulting in relatively large markers. An improved design could be a recursive pattern with a smaller tag nested inside a bigger tag[48] such that a single marker would suffice for use by the UAV from long and short distances. Another approach could be the use of UAV cameras that can zoom allowing for smaller markers to be used. In general, the choice of fiducial marker calls for a balance between detection robustness, computational efficiency, and pose estimation accuracy. There is potential for research into how this compromise can be diminished utilizing more advanced design. Finally, environmental disturbances can pose challenges regarding the overall reliability and robustness of a UAV using markers to land. For instance, strong sustained wind speeds and large wind gusts could overcome the controllability of the UAV regardless of the type of data used (vision- or IMU-based) to estimate the UAV location and pose relative to a landing spot. Also, the robustness of the fiducial markers must be ensured—those worn down by the elements could challenge the detection of the marker.

5 | CONCLUSION

This study explores the development of UAVs as an intelligent agent capable of deploying wireless sensor nodes autonomously for structural health monitoring applications. The proposed UAV system can autonomously detect landing patterns and precisely land upon them to deploy sensors to the structure surface. Transition between different functions like pattern searching and precision landing is accomplished using a reliable finite-state machine embedded in the TX2 onboard computer of the UAVs. Precision positioning of a UAV in the outdoor environment is made possible by the integration of a customized fiducial marker pattern, a robust vision-IMU coupled estimation method, and a discrete Kalman filter. The study reveals sensor positioning accuracy of less than 10 cm. The study also validates UAV sensor deployments for modal analysis. Using two UAV-based wireless sensors, the system can land on a simply supported beam with 5-cm accuracy to extract precise mode shapes. Moving forward, the autonomous UAV sensor delivery system could be improved by incorporating on-the-fly decision-making capabilities. Once structural diagnosis of a certain part of a structure is finished, the system should be able to decide intelligently the next configuration of the sensor network and where it should be installed on the structure. By instantly moving sensors to more advantageous positions on the structure, events of interest such as structural damage can be measured and recorded in a more timely and detailed fashion. Future work also includes expanding the UAV's capability such as to precisely pick up and drop off sensing nodes with an onboard gripper. In this way, fewer UAVs are required to install the sensor network saving deployment cost. Fiducial markers are not always practical in remote outdoor environments. Future work will leverage recent advances in simultaneous localization and mapping (SLAM)[49,50] for autonomous navigation of the UAV in unknown environments.

ACKNOWLEDGEMENTS

This material is based upon the work supported by the National Science Foundation (Grant Numbers 1362975, 1442773, 1446521, and 1831347) and the Office of Naval Research (Grant N00014-21-1-2033). Any opinions, findings, and conclusions or recommendations expressed are only those of the authors.

AUTHOR CONTRIBUTIONS

Hao Zhou: Conceptualization; embedded firmware; system architecture; data collection; analysis; manuscript preparation. **Jerome Lynch:** Conceptualization; system architecture; analysis; manuscript preparation. **Dimitrios Zekkos:** Conceptualization; system architecture; analysis; manuscript preparation.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Jerome Lynch https://orcid.org/0000-0002-8793-0061

REFERENCES

1. Law KH, Lynch JP. Smart city: technologies and challenges. *IT Prof*. 2019;21(6):46-51. doi:10.1109/MITP.2019.2935405
2. Lynch JP. Design of a wireless active sensing unit for localized structural health monitoring. *Struct Control Heal Monit.* 2005;12(3–4): 405-423. doi:10.1002/STC.77
3. Koo KY, Brownjohn JMW, List DI, Cole R. Structural health monitoring of the Tamar suspension bridge. *Struct Control Heal Monit.* 2013;20(4):609-625. doi:10.1002/STC.1481
4. Tang Z, Chen Z, Bao Y, Li H. Convolutional neural network-based data anomaly detection method using multiple information for structural health monitoring. *Struct Control Heal Monit.* 2019;26(1):e2296. doi:10.1002/STC.2296
5. Tsiapoki S, Bahrami O, Häckell MW, Lynch JP, Rolfes R. Combination of damage feature decisions with adaptive boosting for improving the detection performance of a structural health monitoring framework: validation on an operating wind turbine. *Struct Heal Monit*. 2020;20(2):637-660. doi:10.1177/1475921720909379
6. Jang S, Jo H, Cho S, et al. Structural health monitoring of a cable-stayed bridge using smart sensor technology: deployment and evaluation. *Smart Struct Syst*. 2010;6(5_6):461-480.
7. Floreano D, Wood RJ. Science, technology and the future of small autonomous drones. *Nature*. 2015;521(7553):460-466. doi:10.1038/nature14542
8. Greenwood WW, Lynch JP, Zekkos D. Applications of UAVs in civil infrastructure. *J Infrastruct Syst*. 2019;25(2):04019002. doi:10.1061/(asce)is.1943-555x.0000464
9. Sony S, Laventure S, Sadhu A. A literature review of next-generation smart sensing technology in structural health monitoring. *Struct Control Heal Monit.* 2019;26(3):e2321. doi:10.1002/STC.2321
10. Escobar-Wolf R, Oommen T, Brooks CN, Dobson RJ, Ahlborn TM. Unmanned aerial vehicle (UAV)-based assessment of concrete bridge deck delamination using thermal and visible camera sensors: a preliminary analysis. *Res Nondestruct Eval*. 2018;29(4):183-198. doi:10.1080/09349847.2017.1304597
11. Yan Y, Mao Z, Wu J, Padir T, Hajjar JF. Towards automated detection and quantification of concrete cracks using integrated images and lidar data from unmanned aerial vehicles. *Struct Control Heal Monit*. 2021;28(8):e2757. doi:10.1002/STC.2757
12. Hoskere V, Park JW, Yoon H, Spencer BF. Vision-based modal survey of civil infrastructure using unmanned aerial vehicles. *J Struct Eng*. 2019;145(7):04019062. doi:10.1061/(ASCE)ST.1943-541X.0002321
13. Ellenberg A, Kontsos A, Moon F, Bartoli I. Bridge related damage quantification using unmanned aerial vehicle imagery. *Struct Control Heal Monit*. 2016;23(9):1168-1179. doi:10.1002/STC.1831
14. Fujino Y, Siringoringo DM. Recent research and development programs for infrastructures maintenance, renovation and management in Japan. *Struct Infrastruct Eng*. 2020;16(1):3-25. doi:10.1080/15732479.2019.1650077
15. Huston DR, Miller J, Esser B. Adaptive, robotic, and mobile sensor systems for structural assessment. In: Liu S-C, ed. *Smart Structures and Materials 2004: Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems*. Vol. 5391; 2004:189-196.
16. Zhu D, Guo J, Cho C, Wang Y, Lee KM. Wireless mobile sensor network for the system identification of a space frame bridge. *IEEE/ASME Trans Mechatronics*. 2012;17(3):499-507. doi:10.1109/TMECH.2012.2187915
17. Saripalli S, Montgomery JF, Sukhatme GS. Visually guided landing of an unmanned aerial vehicle. *IEEE Trans Robot Autom*. 2003;19(3): 371-380. doi:10.1109/TRA.2003.810239
18. Merz T, Duranti S, Conte G. Autonomous landing of an unmanned helicopter based on vision and inertial sensing. In: *Experimental Robotics IX*. Springer, Berlin/Heidelberg, Germany; 2006:343–352. doi:10.1007/11552246_33
19. Lange S, Sünderhauf N, Protzel P. A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments. In: *2009 International Conference on Advanced Robotics*. IEEE; 2009:1-6.
20. Kato H, Billinghurst M. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In: *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*. 1999:85–94, IEEE. doi:10.1109/IWAR.1999.803809
21. Garrido-Jurado S, Muñoz-Salinas R, Madrid-Cuevas FJ, Marín-Jiménez MJ. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognit*. 2014;47(6):2280-2292. doi:10.1016/j.patcog.2014.01.005
22. Olson E. AprilTag: a robust and flexible visual fiducial system. In: *2011 IEEE International Conference on Robotics and Automation*. 2011:3400–3407, IEEE. doi:10.1109/ICRA.2011.5979561
23. Borowczyk A, Nguyen D-T, Phu-Van Nguyen A, Nguyen DQ, Saussié D, Le Ny J. Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle. *Ifac-Papersonline*. 2017;50(1):10488-10494. doi:10.1016/J.IFACOL.2017.08.1980
24. Chaves SM, Wolcott RW, Eustice RM. NEEC research: toward GPS-denied landing of unmanned aerial vehicles on ships at sea. *Nav Eng J*. 2015;127(1):23-35.
25. Araar O, Aouf N, Vitanov I. Vision based autonomous landing of multirotor UAV on moving platform. *J Intell Robot Syst*. 2017;85(2): 369-384. doi:10.1007/s10846-016-0399-z
26. Rees C. PX4 and 3D Robotics launch new Pixhawk autopilot for UAVs. Published online 2013. https://www.unmannedsystemstechnology.com/2013/08/px4-and-3d-robotics-launch-new-pixhawk-autopilot-for-uavs/
27. MAVLink. MAVLink: micro air vehicle message marshalling library. *GitHub Repos*. Published online 2010. https://github.com/mavlink/mavlink
28. Kane M, Zhu D, Hirose M, Dong X, Winter B, Häckell M, Lynch JP, Wang Y, Swartz A. Development of an extensible dual-core wireless sensing node for cyber-physical systems. In: *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2014*. Vol 9061, International Society for Optics and Photonics; 2014. doi:10.1117/12.2045325

29. ArduPilot. ArduPilot project. *GitHub Repos*. Published online 2011. https://github.com/ArduPilot/ardupilot
30. ArduPilot. ArduPilot EKF. Published online 2011. https://ardupilot.org/copter/docs/common-apm-navigation-extended-kalman-filter-overview.html
31. Pittelkau ME. Rotation vector in attitude estimation. *J Guid Control Dynam*. 2003;26(6):855-860. doi:10.2514/2.6929
32. DroneKit. DroneKit-Python: library for communicating with drones via MAVLink. Published online 2014. https://github.com/dronekit/dronekit-python
33. Swatbotics. Apriltag. *GitHub Repos*. Published online 2016. https://github.com/swatbotics/apriltag
34. Zhou H, Lynch JP, Zekkos D. Vision-based precision localization of UAVs for sensor payload placement and pickup for field monitoring applications. In: *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2019*. Vol. 10970. SPIE; 2019: 1097007.
35. Lu CP, Hager GD, Mjolsness E. Fast and globally convergent pose estimation from video images. *IEEE Trans Pattern Anal Mach Intell*. 2000;22(6):610-622. doi:10.1109/34.862199
36. Lepetit V, Moreno-Noguer F, Fua P. EPnP: an accurate $O(n)$ solution to the PnP problem. *Int J Comput Vis*. 2009;81(2):155-166. doi:10.1007/s11263-008-0152-6
37. Bradski G. The OpenCV library. *Dr Dobb's J Softw Tools*. 2000;25:120-125.
38. OpenCV. Camera calibration and 3D reconstruction, OpenCV 2.4.13.7 documentation. Published online 2019.
39. Levenberg K. A method for the solution of certain non-linear problems in least squares. *Q Appl Math*. 1944;2(2):164-168. doi:10.1090/qam/10666
40. Marquardt DW. An algorithm for least-squares estimation of nonlinear parameters. *J Soc Ind Appl Math*. 1963;11(2):431-441. doi:10.1137/0111030
41. Leutenegger S, Lynen S, Bosse M, Siegwart R, Furgale P. Keyframe-based visual–inertial odometry using nonlinear optimization. *Int J Rob Res*. 2015;34(3):314-334. doi:10.1177/0278364914554813
42. Forster C, Carlone L, Dellaert F, Scaramuzza D. On-manifold preintegration for real-time visual-inertial odometry. *IEEE Trans Robot*. 2016;33(1):1-21. doi:10.1109/TRO.2016.2597321
43. Qin T, Li P, Shen S. VINS-mono: a robust and versatile monocular visual-inertial state estimator. *IEEE Trans Robot*. 2018;34(4): 1004-1020. doi:10.1109/TRO.2018.2853729
44. Mourikis AI, Roumeliotis SI. A multi-state constraint Kalman filter for vision-aided inertial navigation. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE; 2007:3565-3572.
45. Bloesch M, Omari S, Hutter M, Siegwart R. Robust visual inertial odometry using a direct EKF-based approach. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015:298–304, IEEE. doi:10.1109/IROS.2015.7353389
46. Bar-Shalom Y, Li XR, Kirubarajan T. *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. New York, NY: John Wiley & Sons; 2004.
47. Brincker R, Zhang L, Andersen P. Modal identification from ambient responses using frequency domain decomposition. In: *IMAC 18: Proceedings of the International Modal Analysis Conference (IMAC)*; 2000:625-630.
48. Krogius M, Haggenmiller A, Olson E. Flexible layouts for fiducial tags. In: *IEEE International Conference on Intelligent Robots and Systems*. IEEE; 2019:1898-1903.
49. Durrant-Whyte H, Bailey T. Simultaneous localization and mapping: part I. *IEEE Robot Autom Mag*. 2006;13(2):99-110. doi:10.1109/MRA.2006.1638022
50. Bailey T, Durrant-Whyte H. Simultaneous localization and mapping (SLAM): part II. *IEEE Robot Autom Mag*. 2006;13(3):108-117. doi:10.1109/MRA.2006.1678144

How to cite this article: Zhou H, Lynch J, Zekkos D. Autonomous wireless sensor deployment with unmanned aerial vehicles for structural health monitoring applications. *Struct Control Health Monit*. 2022;29(6):e2942. doi:10.1002/stc.2942