NEEC Research: Toward GPS-denied Landing of Unmanned Aerial Vehicles on Ships at Sea

Stephen M. Chaves, Ryan W. Wolcott, and Ryan M. Eustice

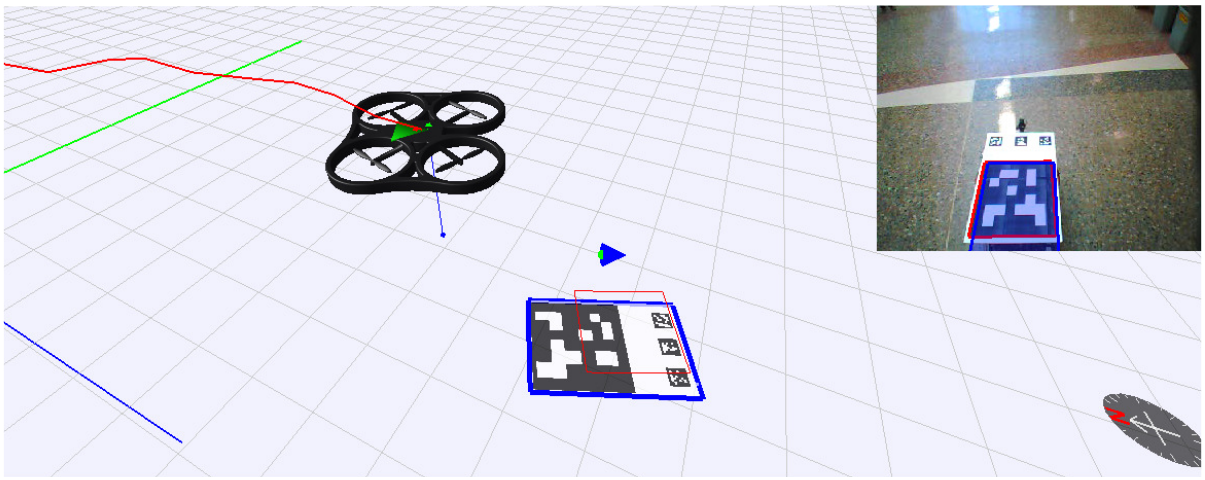

Fig. 1: Real-time visualization of the proposed system for autonomously landing a UAV on a transiting ship. The UAV estimates its relative-pose to the landing platform through observations of fiducial markers and an EKF. Our visualization displays the UAV trajectory, control commands, waypoints, fiducial marker detections, estimated poses of the UAV and landing platform, and an embedded stream of the UAV's onboard camera images.

***Abstract*—This paper reports on a Naval Engineering Education Center (NEEC) design-build-test project focused on the development of a fully autonomous system for landing Navy unmanned aerial vehicles (UAVs) on transiting ships at sea. Our NEEC team of engineering students researched image processing techniques, estimation frameworks, and control algorithms to collaboratively learn and train in Navy-relevant autonomy. We accomplished the autonomous landing using fiducial markers detected by a camera onboard the UAV, decomposed the resulting homography into 6-degree of freedom relative-pose information between the UAV and landing platform, performed state estimation with a delayed-state extended Kalman filter, and then executed the motion control. Results for the state estimation framework and experimental tests using a motion capture system for independent ground-truth are presented. The proposed system performed successfully on a robotic testbed consisting of a micro UAV and an unmanned ground vehicle.**

Keywords: ***UAV, autonomy, future Navy, robotics, estimation, computer vision, image processing*.**

I. Introduction

Unmanned autonomous systems are playing an ever-growing and more crucial role in the Navy active fleet [1], [2]. As such, the U.S. Navy has a need to investigate the science and technology behind the next-generation of unmanned autonomous systems, and to educate and hire a cross-disciplinary civilian workforce trained in this area. In this paper, we report on a Naval Engineering Education Center (NEEC) project that is helping to fill that role by educating and training an undergraduate and graduate student cohort in the cross-disciplinary issues in developing robust autonomy for unmanned vehicle systems. We focus the content of this paper on the project research and tools used for training our team of students.

During the 2011-2012 academic year, our team of students investigated image processing techniques, estimation frameworks, and control algorithms in order to develop a new system for autonomously landing an unmanned aerial vehicle (UAV) on the deck of a Navy ship, regarded as one of the most difficult autonomous tasks for military aviation [3], [4].

Currently, the active duty UAVs in use by the Navy rely on systems that communicate between the UAV and the ship. One such method used by the Northrop Grumman FireScout UAV employs a ship-mounted millimeter-wave radar and corresponding transponder to guide the UAV during the landing [5]. Another method that depends on vehicle-to-ship communcation is to use a local ship-borne real-time kinematic (RTK) global positioning system (GPS) and UAV-to-ship communications for precise localization and command execution, as with the recently-developed Northrop Grumman X-47B [6]. Autonomous landing tests by the X-47B onto active-duty aircraft carriers are already in progress, however, these methods rely on systems external to the UAV itself, and

Manuscript received April 2013. This work was supported by the Naval Sea Systems Command through the Naval Engineering Education Center (NEEC) under contract number N65540-10-C-003.

The authors are with the Perceptual Robotics Laboratory at the University of Michigan. S. Chaves is a PhD student in the Dept. of Mechanical Engineering, R. Wolcott is a PhD student in the Dept. of Computer Science, and R. Eustice is an Associate Professor in the Dept. of Naval Architecture and Marine Engineering.

can be jammed, scrambled, or spoofed [7]. Thus, a more robust method for autonomous landing is desired, without sacrificing any of the localization accuracy provided by GPS or radar.

There has been significant previous work on the topic of GPS-denied autonomous landings of UAVs on ships. Oh et. al. proposed using a tether between the UAV and ship to aid in tracking the ship's motion [8]. Garratt et al. combined an optical beacon with a laser rangefinder to create a passive landing system that does not rely on a vehicle-to-ship connection [9]. A method based on visual servoing is given by Coutard et al. in [4], but requires a continuous, uninterrupted view of visual features on the ship deck. Herisse et al. developed a visual system based on optical flow measurements, but assume the normal of the landing platform is known [10]. Recently, Arora et al. demonstrated a lidar and camera-based shipdeck tracking system on a full-size helicopter [11]. The work closest to ours in terms of proposed system and experimental demonstrations is that of Richardson et al. [12], who independently and in parallel developed a visual tracking system using known markers to estimate the relative-pose between the UAV and landing platform. Their work focuses heavily on the control system design and testing against disturbances like wind gusts and lighting variation.

In this paper, we propose a system for autonomous UAV landings that is centered on computer vision techniques and a delayed-state estimation framework. The system runs entirely onboard the UAV and no communication to it is necessary to perform the landing. It relies on no external systems, like GPS, and features only the UAV's onboard processors and sensors, and a marked landing platform on the ship deck. Further, the state estimator provides robustness to interrupted views of the landing platform.

A. Conceptual Overview

Conceptually, the proposed system for autonomous landing is straightforward. A camera is mounted on the UAV in such a way as to view the landing platform on the ship during approach and descent. (On our experimental platform, this was a single forward-looking camera angled slightly downward.) The only component of the system present on the Navy ship is one or more fiducial markers that designate the landing area on the ship's deck. The number, size, and visual attributes of markers are application-dependent. They should clearly outline the landing area, but generally be within the field of view of the UAV's onboard camera throughout the landing process, although the state estimator can account for temporary occlusions or interruptions in view.

On the UAV itself, the system is composed of an inertial measurement unit (IMU), the onboard camera, and a central processor for running all algorithms and handling inter-process communication. The system uses the camera's video stream to detect the fiducial markers on the landing platform. Then, the image processing algorithm uses these detections to extract relative-pose information between the UAV and the landing target. This information is fused with measurements from the UAV's inertial sensor in a delayed-state extended Kalman filter (EKF) to give a more accurate state estimate and real-time operation. Using the localization estimate, the UAV executes the autonomous landing maneuver. Fig. 1 provides a visualization of the system in operation.

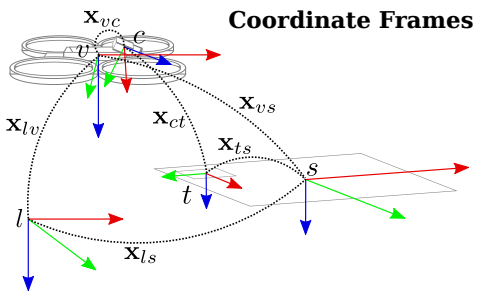

Fig. 2: View of example coordinate frames and pose relations for the proposed landing system. $\mathbf{x}_{lv}$ is the pose of the vehicle with respect to the local frame, $\mathbf{x}_{ls}$ is the pose of the ship with respect to the local frame, $\mathbf{x}_{vc}$ is the pose from vehicle to camera, $\mathbf{x}_{ct}$ is the pose from camera to tag, $\mathbf{x}_{ts}$ is the pose from tag to ship, and $\mathbf{x}_{vs}$ is the pose from vehicle to ship.

With this design, the system is purely passive and does not depend at all upon outside communication. The fiducial marker detections allow the system to estimate the full 6-degree of freedom (DOF) pose information of the UAV and ship, enabling operation in rough sea states with a significantly-pitching and rolling shipdeck.

II. Background Information

Here we present some necessary background information for understanding the operation of the proposed autonomous landing system. The system relies on relative-pose measurements originating from the camera onboard the UAV relating the UAV pose to the ship pose. Thus, it is important to understand the coordinate frames used and the appropriate mathematical operations for correctly transforming information between frames.

A. Coordinate Frames

The UAV frame follows right-handed z-down convention such that the positive x-axis is oriented along the UAV's forward direction of travel, and the frame is centered and aligned with the axes of the onboard IMU. The camera frame is fixed with respect to the UAV frame, but translated and rotated such that the positive z-axis points out of the camera lens. The x-axis points to the right from the image center and the y-axis points down, forming a right-handed frame.

The ship frame also follows right-handed z-down convention and is positioned at the center of the landing platform. Finally, we define a North-East-Down (NED) local frame that is fixed with respect to the world and initialized by the system at an arbitrary location. Refer to Fig. 2 for a depiction of the system coordinate frames.

We can now define the pose of frame j with respect to frame i as the 6-DOF vector

$$\mathbf{x}_{ij} = [{}^i\mathbf{t}_{ij}^\top, \mathbf{\Theta}_{ij}^\top]^\top = [x_{ij}, y_{ij}, z_{ij}, \phi_{ij}, \theta_{ij}, \psi_{ij}]^\top, \quad (1)$$

composed of the translation vector from frame i to frame j as defined in frame i and the roll, pitch, and heading Euler angles.

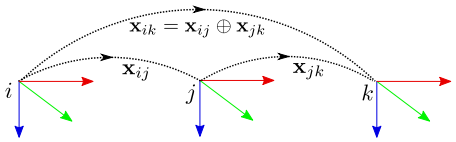

Fig. 3: The head-to-tail operation of coordinate frame poses.

Then, the homogeneous coordinate transformation from frame j to frame i is written as

$$ {}^{i}_{j}\mathrm{H} = \begin{bmatrix} {}^{i}_{j}\mathrm{R} & {}^{i}\mathbf{t}_{ij}^{\top} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2} $$

where ${}^{i}_{j}\mathrm{R}$ is the orthonormal rotation matrix that rotates frame j into frame i and is defined as

$$ {}^{i}_{j}\mathrm{R} = \mathrm{rotxyz}(\boldsymbol{\Theta}_{ij}) = \mathrm{rotz}(\psi_{ij})^{\top}\mathrm{roty}(\theta_{ij})^{\top}\mathrm{rotx}(\phi_{ij})^{\top}. \tag{3} $$

B. Coordinate Frame Operations

We present here coordinate frame operations that are necessary for relating pose estimates to useful information about the UAV and the ship deck. Further details about coordinate frames and operations can be found in [13].

1) Head-to-Tail Operation: The compounding of two frames is described by the head-to-tail operation (see Fig. 3), which is defined as $\mathbf{x}_{ik} = \mathbf{x}_{ij} \oplus \mathbf{x}_{jk}$, with corresponding homogeneous coordinate transform from frame k to frame i given by ${}^{i}_{k}\mathrm{H} = {}^{i}_{j}\mathrm{H}\,{}^{j}_{k}\mathrm{H}$. Given random variables $\mathbf{x}_{ij}$ and $\mathbf{x}_{jk}$, the Jacobian for the head-to-tail is denoted as

$$ \mathrm{J}_{\oplus} = \frac{\partial \mathbf{x}_{ik}}{\partial(\mathbf{x}_{ij}, \mathbf{x}_{jk})} = \begin{bmatrix} \mathrm{J}_{\oplus 1} & \mathrm{J}_{\oplus 2} \end{bmatrix}. \tag{4} $$

2) Inverse Operation: Reversing a relationship between coordinate frames is described by the inverse operation and is denoted as $\mathbf{x}_{ji} = \ominus\mathbf{x}_{ij}$, where the homogeneous coordinate transform is simply ${}^{j}_{i}\mathrm{H} = {}^{i}_{j}\mathrm{H}^{-1}$. The Jacobian of the inverse operation is denoted as

$$ \mathrm{J}_{\ominus} = \frac{\partial \mathbf{x}_{ji}}{\partial \mathbf{x}_{ij}}. \tag{5} $$

3) Tail-to-Tail Operation: The tail-to-tail operation describes the relationship between two coordinate frames that are already expressed in a common frame. In this case, the tail-to-tail operation is composed of an inverse operation followed by a head-to-tail operation, as denoted as

$$ \mathbf{x}_{jk} = \ominus\mathbf{x}_{ij} \oplus \mathbf{x}_{ik} \tag{6} $$

$$ = \mathbf{x}_{ji} \oplus \mathbf{x}_{ik}. \tag{7} $$

As expected, the homogeneous coordinate transform for the tail-to-tail operation is given by ${}^{j}_{k}\mathrm{H} = {}^{j}_{i}\mathrm{H}\,{}^{i}_{k}\mathrm{H} = {}^{i}_{j}\mathrm{H}^{-1}\,{}^{i}_{k}\mathrm{H}$ and the Jacobian is found by

$$ {}_{\ominus}\mathrm{J}_{\oplus} = \frac{\partial \mathbf{x}_{jk}}{\partial(\mathbf{x}_{ij}, \mathbf{x}_{ik})} \tag{8} $$

$$ = \frac{\partial \mathbf{x}_{jk}}{\partial(\mathbf{x}_{ji}, \mathbf{x}_{ik})} \cdot \frac{\partial(\mathbf{x}_{ji}, \mathbf{x}_{ik})}{\partial(\mathbf{x}_{ij}, \mathbf{x}_{ik})} \tag{9} $$

$$ = \begin{bmatrix} \mathrm{J}_{\oplus 1}\mathrm{J}_{\ominus} & \mathrm{J}_{\oplus 2} \end{bmatrix}. \tag{10} $$

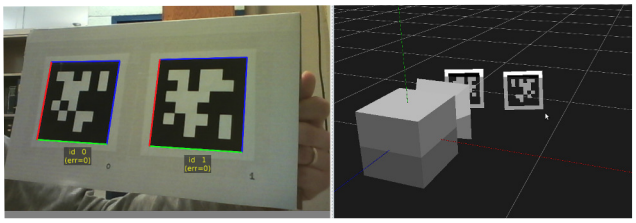

Fig. 4: Sample AprilTag fiducial markers, corresponding detection boxes, and a view of the relative-poses from camera to tags. We use AprilTags [14] in our system to simplify the image processing task of detecting the landing platform.

III. System Description

We now present a full description of the proposed autonomous landing system, divided into three main modules, namely (*i*) image processing, (*ii*) state estimation, and (*iii*) motion control. These three modules together define the core methodology of the design, and are presented in detail below.

A. Image Processing

The proposed system hinges upon the detection of fiducial markers by the UAV's onboard camera in order to perform the autonomous landing. The fiducial markers serve as information-rich features from which the UAV extracts the relative-pose of the landing platform to feed to the state estimation framework.

The fiducial detection system employed by our experimental testbed (described later) is called AprilTag [14]. The AprilTag markers (see Fig. 4) are high-contrast 2D tags, similar to QR codes, but designed to be robust to low image resolution, occlusions, rotations, and lighting variation. For robustness, these tags should be asymmetric and composed of high-contrast colors to facilitate detection by the UAV's cameras. Tag detection in an image begins by analyzing each pixel's gradient direction and magnitude and clustering pixels by similar gradient characteristics. From the clustering, directed line segments are filtered out and examined for formations of candidate markers.

With points from a candidate marker, the Direct Linear Transform (DLT) algorithm [15] solves the homography mapping the tag coordinate frame to the camera frame, ${}^{t}_{c}\mathrm{H}$, and the candidate is compared for a match against a database of pre-trained markers. Using *a priori* calibration information of the camera's focal length and the actual size of the fiducial marker of interest, we calculate the 6-DOF relative-pose of the tag frame with respect to the UAV camera frame, i.e. $\mathbf{x}_{ct}$, which is then fed to the state estimation framework as an observation.

It is worth noting that other types of pre-trained fiducial markers can be used, as in [12].

B. State Estimation

State estimation for the proposed system is accomplished in a delayed-state EKF framework [16]. The filter estimates and tracks both the UAV and landing platform poses with respect to the local frame, which is arbitrarily initialized at start.

1) UAV Estimation: At first, the state vector in the EKF is composed of only the UAV (*v*) pose relative to the local frame (*l*), as well as its velocities and rates. The state vector is written as the 12-element vector $\mathbf{x}_{KF}$ as

$$\mathbf{x}_v = [\mathbf{x}_{lv}^\top, {}^l\underline{\mathbf{v}}_v^\top, {}^l\underline{\mathbf{w}}_v^\top]^\top, \tag{11}$$

$$\mathbf{x}_{KF} = \mathbf{x}_v \tag{12}$$

where ${}^l\underline{\mathbf{v}}_v = [u, v, w]^\top$ and ${}^l\underline{\mathbf{w}}_v = [p, q, r]^\top$ are the local-frame velocities and angular velocities, respectively. We employ a simple constant velocity process model with white Gaussian noise induced on the velocity terms for the UAV, given by

$$\bar{\mathbf{x}}_{KF_k} = \mathrm{F}_{k-1}\mathbf{x}_{KF_{k-1}} + \mathbf{w}_{k-1}, \tag{13}$$

with process noise $\mathbf{w}_{k-1} \sim \mathcal{N}(0, \mathrm{Q}_{k-1})$ and $\mathrm{F}_{k-1} = \mathrm{F}_{\mathbf{x}_v}$ where

$$\mathrm{F}_{\mathbf{x}_v} = \begin{bmatrix} I_{6\times6} & \Delta t \cdot I_{6\times6} \\ 0_{6\times6} & I_{6\times6} \end{bmatrix}. \tag{14}$$

2) Ship Initialization: Upon first observation of the landing platform from the camera, we augment the state vector to include the local-frame pose, velocities, and rates of the ship (*s*). Now, the EKF state vector is written as the 24-element vector $\mathbf{x}_{KF} = [\mathbf{x}_v^\top, \mathbf{x}_s^\top]^\top$ with

$$\mathbf{x}_v = [\mathbf{x}_{lv}^\top, {}^l\underline{\mathbf{v}}_v^\top, {}^l\underline{\mathbf{w}}_v^\top]^\top, \tag{15}$$

$$\mathbf{x}_s = [\mathbf{x}_{ls}^\top, {}^l\underline{\mathbf{v}}_s^\top, {}^l\underline{\mathbf{w}}_s^\top]^\top. \tag{16}$$

To remain consistent with the assumption of the ship's strictly planar dynamics, we use a planar constant velocity model for the ship, where white Gaussian noise is added to the ship's planar velocity terms, such that its linear process model is

$$\mathrm{F}_{\mathbf{x}_s} = \begin{bmatrix} I_{2\times2} & 0_{2\times4} & \Delta t \cdot I_{2\times2} & 0_{2\times4} \\ 0_{4\times2} & I_{4\times4} & 0_{4\times2} & 0_{4\times4} \\ 0_{2\times2} & 0_{2\times4} & I_{2\times2} & 0_{2\times4} \\ 0_{4\times2} & 0_{4\times4} & 0_{4\times2} & I_{4\times4} \end{bmatrix} \tag{17}$$

and the process model for the full 24-element state vector is the block diagonal matrix

$$\mathrm{F}_{k-1} = \begin{bmatrix} \mathrm{F}_{\mathbf{x}_v} & 0_{12\times12} \\ 0_{12\times12} & \mathrm{F}_{\mathbf{x}_s} \end{bmatrix}. \tag{18}$$

3) Sensor Observation Models: Two types of sensor observations are included in the EKF formulation: observations from the UAV's IMU and camera observations forwarded by the image processing algorithm. The generic observation model is described by the nonlinear (but differentiable) function

$$\mathbf{z}_x[k] = \mathbf{h}_x(\mathbf{x}_{KF_k}) + \mathbf{v}_x[k], \tag{19}$$

with observation noise $\mathbf{v}_x[k] \sim \mathcal{N}(0, \mathrm{R}_x[k])$, where x denotes the sensor, either *IMU* or *tag*.

The IMU configuration used in our experimental testbed does some low-level processing before returning observations, so the IMU actually reports a sensor packet that contains UAV altitude, Euler angles, and horizontal body-aligned *x* and *y* velocities. As such, the IMU sensor model is given as

$$\mathbf{h}_{IMU}(\mathbf{x}_{KF}) = \begin{bmatrix} z_{lv} \\ \phi_{lv} \\ \theta_{lv} \\ \psi_{lv} \\ u_{lv}\cos\psi_{lv} + v_{lv}\sin\psi_{lv} \\ v_{lv}\cos\psi_{lv} - u_{lv}\sin\psi_{lv} \end{bmatrix}, \tag{20}$$

with Jacobian

$$\mathrm{H}_{IMU} = \frac{\partial \mathbf{h}_{IMU}}{\partial \mathbf{x}_{KF}} = \begin{bmatrix} \frac{\partial \mathbf{h}_{IMU}}{\partial \mathbf{x}_v} & \frac{\partial \mathbf{h}_{IMU}}{\partial \mathbf{x}_s} \end{bmatrix} \tag{21}$$

$$= \begin{bmatrix} \frac{\partial \mathbf{h}_{IMU}}{\partial \mathbf{x}_v} & 0_{6\times12} \end{bmatrix}. \tag{22}$$

From the state vector, we have estimates of both the UAV pose, $\mathbf{x}_{lv}$, and ship pose, $\mathbf{x}_{ls}$, with respect to the local frame. The static coordinate transforms $\mathbf{x}_{vc}$ and $\mathbf{x}_{st_i}$ denote the pose of the camera frame relative to the UAV frame and the i^{th} tag coordinate frame relative to the ship, respectively. Thus, we can compute the relative-pose of an individual tag with respect to the camera from a series of coordinate frame operations, resulting in the sensor model for individual tag observations,

$$\mathbf{h}_{tag_i}(\mathbf{x}_{KF}) = \mathbf{x}_{ct_i} = (\ominus\mathbf{x}_{vc} \oplus (\ominus\mathbf{x}_{lv} \oplus \mathbf{x}_{ls})) \oplus \mathbf{x}_{st_i} \tag{23}$$

with Jacobian H_{tag}.

4) Discrete Extended Kalman Filter: The state vector input to the extended Kalman filter is assumed to be normally distributed with mean and covariance as $\mathbf{x}_{KF} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, where the covariance blocks are written out as

$$\Sigma = \begin{bmatrix} \Sigma_{\mathbf{x}_v} & \Sigma_{\mathbf{x}_v\mathbf{x}_s} \\ \Sigma_{\mathbf{x}_s\mathbf{x}_v} & \Sigma_{\mathbf{x}_s} \end{bmatrix}. \tag{24}$$

We define a discrete-time process model and include nonlinear discrete sensor observations, so the estimation framework is written as a discrete first-order EKF with the following prediction and update steps.

$$\begin{aligned} \text{Predict} \quad & \bar{\boldsymbol{\mu}}_k = \mathrm{F}_{k-1}\boldsymbol{\mu}_{k-1} \\ & \bar{\Sigma}_k = \mathrm{F}_{k-1}\Sigma_{k-1}{\mathrm{F}_{k-1}}^\top + \mathrm{Q}_{k-1} \\ \\ \text{Update} \quad & \mathrm{K}_k = \bar{\Sigma}_k{\mathrm{H}_x}^\top(\mathrm{H}_x\bar{\Sigma}_k{\mathrm{H}_x}^\top + \mathrm{R}_x[k])^{-1} \\ & \boldsymbol{\mu}_k = \bar{\boldsymbol{\mu}}_k + \mathrm{K}_k(\mathbf{z}_x[k] - \mathbf{h}_x(\bar{\boldsymbol{\mu}}_k)) \\ & \Sigma_k = (\mathrm{I} - \mathrm{K}_k\mathrm{H}_x)\bar{\Sigma}_k(\mathrm{I} - \mathrm{K}_k\mathrm{H}_x)^\top + \mathrm{K}_k\mathrm{R}_x[k]{\mathrm{K}_k}^\top \end{aligned} \tag{25}$$

In these equations, $\bar{\mathbf{x}}_{KF_k} \sim \mathcal{N}(\bar{\boldsymbol{\mu}}_k, \bar{\Sigma}_k)$ represents the predicted state vector at time *k*, K_k represents the Kalman gain, F_{k-1} represents the process model, and H_x represents the observation model Jacobian. During an update step, the Jacobian is evaluated at the predicted state estimate and applied to the covariance. We use the Joseph form of the covariance update equation to ensure a positive-definite covariance [17].

5) Delayed-State Formulation: Because of the latency present in the camera image pipeline, an observation of the fiducial markers on the landing platform is received by the state estimation framework at a time later than the time of image capture. This means that the current camera observation does not directly correspond to the current state estimate, but rather corresponds to a state estimate in the past. On our experimental testbed, latencies for incorporating camera measurements were on the order of several hundred milliseconds. Most of this delay was due to streaming data to and from the micro UAV and the processing laptop. The image processing module itself has a computation time on the order of tens of milliseconds.

No matter the duration of the latency, the framework for the EKF must be modified to accommodate these delayed observations [16]. We handle this modification by augmenting a history of UAV and ship poses into the state estimate such that the state vector is

$$\mathbf{x}_{DS_k} = [\mathbf{x}_{KF_k}^\top, \mathbf{x}_{KF_{k-1}}^\top, \ldots, \mathbf{x}_{KF_{k-n}}^\top]^\top \tag{26}$$

where n is the number of delayed-states.

We then augment the rest of the EKF framework to handle the delayed-state formulation. The constant velocity process model continues to act on only the current UAV and ship poses, so the process model becomes

$$\bar{\mathbf{x}}_{DS_k} = \begin{bmatrix} \bar{\mathbf{x}}_{KF_k} \\ \bar{\mathbf{x}}_{KF_{k-1}} \\ \vdots \\ \bar{\mathbf{x}}_{KF_{k-n}} \end{bmatrix} = \begin{bmatrix} \mathrm{F}_{k-1}\mathbf{x}_{KF_{k-1}} + \mathbf{w}_{k-1} \\ \mathbf{x}_{KF_{k-1}} \\ \vdots \\ \mathbf{x}_{KF_{k-n}} \end{bmatrix}. \tag{27}$$

Similarly, the IMU sensor model Jacobian becomes

$$\mathrm{H}_{DS_{IMU}} = \begin{bmatrix} \mathrm{H}_{IMU} & 0_{6\times(24\cdot n)} \end{bmatrix}. \tag{28}$$

Lastly, we must modify the tag observation model to account for the delayed measurement received from the image processing algorithm. Since these measurements are a function of the delayed-states, the observation model corresponding to the image captured at the time of delayed-state k is given by

$$\mathbf{h}_{DS_{tag_i}}(\mathbf{x}_{DS}) = \mathbf{x}_{ct_i}[k] = (\ominus\mathbf{x}_{vc} \oplus (\ominus\mathbf{x}_{lv}[k] \oplus \mathbf{x}_{ls}[k])) \oplus \mathbf{x}_{st_i} \tag{29}$$

with corresponding Jacobian $\mathrm{H}_{DS_{tag}}$.

After the Kalman filter update is performed for each delayed observation, we marginalize the corresponding delayed-state from the EKF, as well as any previous delayed-states. This is accomplished by striking the associated entries in the state vector and covariance matrix. In this way, the filter carries only the current state and a short history of delayed-states with not-yet-accounted-for image measurements. Further, we limit the number of delayed-states to reduce the overall state size in the EKF.

The state estimator tracks the UAV and ship local-poses in order to initialize and process observations easily, but the relative-pose from UAV to ship, i.e. $\mathbf{x}_{vs}$, is the actual quantity of interest for the rest of our system. The relative-pose is derived from the state vector as

$$\mathbf{x}_{vs} = \ominus\mathbf{x}_{lv} \oplus \mathbf{x}_{ls} \tag{30}$$

with corresponding covariance calculated using the tail-to-tail Jacobian,

$$\Sigma_{\mathbf{x}_{vs}} = {}_{\ominus}\mathrm{J}_{\oplus} \begin{bmatrix} \Sigma_{\mathbf{x}_v} & \Sigma_{\mathbf{x}_v\mathbf{x}_s} \\ \Sigma_{\mathbf{x}_s\mathbf{x}_v} & \Sigma_{\mathbf{x}_s} \end{bmatrix} {}_{\ominus}\mathrm{J}_{\oplus}{}^\top. \tag{31}$$

From this formulation, the state estimation framework returns an accurate, filtered, continuous real-time estimate of the relative-pose and uncertainty from the UAV to the ship from which the motion control module executes the autonomous landing.

C. Motion Control

The motion control module is the final component of the overall methodology of the proposed system. The motion controller is responsible for the planning and execution of the landing event. We subdivide the motion control module into a mission manager (described later) and two controllers – a low-level attitude controller to stabilize the vehicle and a high-level position controller for trajectory-following during descent and landing. In our experimental platform, the low-level attitude controller is designed by the UAV manufacturer, so our motion control design is focused on the high-level position controller.

We first define the descent trajectory to accommodate certain performance and safety requirements. We design the descent of the UAV toward the landing platform to always approach from the stern of the ship, assuming forward ship motion at sea. In addition, the descent trajectory is designed to provide the UAV camera with a continuous view of the fiducial markers on the landing platform throughout the entire landing event, despite the system being robust to interruptions in view. Thus, the descent trajectory roughly follows a straight-line path traced from the camera lens to the landing platform. For best performance, the UAV should follow the planned trajectory with minimal deviations, so as to maximize the number of fiducial marker observations on the platform. To do so, we use a pure pursuit path follower [18] which guides the UAV along a line connecting waypoints in the planned trajectory.

To execute the landing maneuver, the high-level position controller employs proportional-integral-derivative (PID) feedback loops around x, y, z, and yaw error signals in the UAV body-frame. Error signals are calculated by transforming the next trajectory waypoint from the local frame (l) to the current UAV body frame (v).

For added robustness, our system continually evaluates the uncertainty surrounding the relative-pose from UAV to landing platform and determines whether the confidence in the estimate is high enough to perform a safe landing. We evaluate the sixth-root of the determinant of $\Sigma_{\mathbf{x}_{vs}}$ as a measure of the total uncertainty of the relative-pose [19] and compare this value to a confidence threshold for landing. If the uncertainty reaches above the threshold, the system instructs the state estimator to marginalize out state vector elements corresponding to the ship and re-initialize upon the next camera observation. The UAV is also instructed to back off and re-attempt the landing once the ship has been detected again and the confidence is higher. This check adds a layer of robustness to the system to prevent unsafe landing attempts.

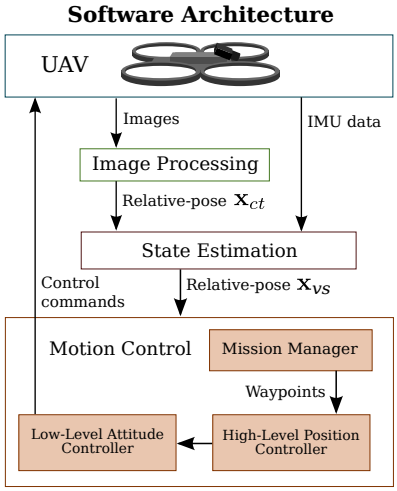

Fig. 5: The proposed system software architecture. Arrows between processes represent LCM communications.

IV. Software Architecture

This section describes the software architecture of the proposed system such that it enables organization of the overall approach into the three modules outlined previously. Fig. 5 shows the layout of the software architecture.

1) Inter-Process Communication: Before we discuss the specific processes that comprise the modules, we must first identify the format and protocol for inter-process communication within the architecture. For message passing between processes, we use LCM [20]. LCM is a publish/subscribe message passing service that was specifically designed for real-time applications in robotics and autonomy. It is robust, language-independent, and facilitates the setup of a modular software architecture, as it inherently creates message-passing boundaries between processes and standardizes the message format within the architecture.

LCM also features some built-in utilities that are beneficial for software development and analysis, depicted in Fig. 6. LCM Logger and LogPlayer allow recording and playing-back LCM network traffic. LCM Spy is a tool for displaying LCM messages in real-time. These utilities are handy when debugging code, but are especially beneficial for development; we are able to play-back log files of real-world sensor data or outputs from certain software modules without actually running the sensors or modules themselves.

2) Image Processing: Within the software architecture, the image processing module is comprised of a camera driver and the AprilTag fiducial marker detection system. This module directly interfaces to the camera onboard the UAV and publishes an image stream from the camera and the resulting camera-to-tag poses over LCM.

3) State Estimation: The state estimation module runs the delayed-state extended Kalman filter and subscribes to observations published directly from the UAV (in the case of IMU data) and from the image processing module (for camera observations). This module publishes the derived relative-pose estimate of the ship with respect to the UAV.

4) Mission Manager: The mission manager is a high-level decision-maker embedded within the motion control module that serves as an interface to the motion controllers and guides the UAV through the many stages of the autonomous landing process. The mission manager subscribes to the relative-pose $\mathbf{x}_{vs}$ from the state estimator to calculate the descent trajectory. As the UAV flies, the mission manager acts as a state machine, cycling through phases of the descent and publishing waypoints using the pure pursuit path follower. The mission manager also governs the sending of commands for events like engine shutdown after completing the landing. Each phase of the descent can be encoded with operational or performance requirements that correspond to the progress of the UAV as it executes the landing maneuver. The confidence threshold check, for example, is handled by the mission manager.

5) Motion Controllers: As stated previously, motion control of the UAV is accomplished using two separate controllers – one for attitude stabilization and one for position control. Both controllers subscribe to the desired trajectory waypoints published by the mission manager and the relative-pose vector $\mathbf{x}_{vs}$ published by the EKF. Only the attitude controller directly commands the UAV's motion, as the UAV moves by varying its thrust on individual motors to obtain a desired orientation.

V. Experiments and Results

We tested the proposed system by autonomously landing a micro UAV on an unmanned ground vehicle (UGV). This testbed serves as a proxy for a full-scale setup and is meant to conceptually demonstrate the viability of the proposed system.

A. Robotic Platform

The micro UAV we used for experimental testing was a Parrot AR.Drone [21]. The AR.Drone is a quadrotor helicopter that features an internal micro-electro-mechanical system (MEMS) IMU, ultrasonic altimeter, and forward-looking and downward-looking cameras. We angled the forward-looking camera 45° downward to provide a field of view that facilitated landing on the UGV. Only the modified forward-looking camera was used for landing platform detection.

The AR.Drone does not have any significant onboard processing capability to host the software for the proposed system. Instead, we used a laptop computer to run the system code and stream commands to the UAV over a wireless connection. The UAV streamed IMU data and camera images to the laptop over this connection as well. (Note: streaming communications are not meant to be included in the final system that runs completely onboard the UAV, but are necessary here because of limitations with our experimental setup.)

The UGV was a Segway RMP-200 [22] equipped with a landing platform and four AprilTag markers – one large marker for initial detection during the early phases of the descent, and three smaller markers for fine pose control when completing the landing maneuver (markers can be seen in Fig. 1). The Segway itself had no communication to the AR.Drone or laptop computer and was piloted via remote-control by a human operator. The experimental platform is shown in Fig. 7.

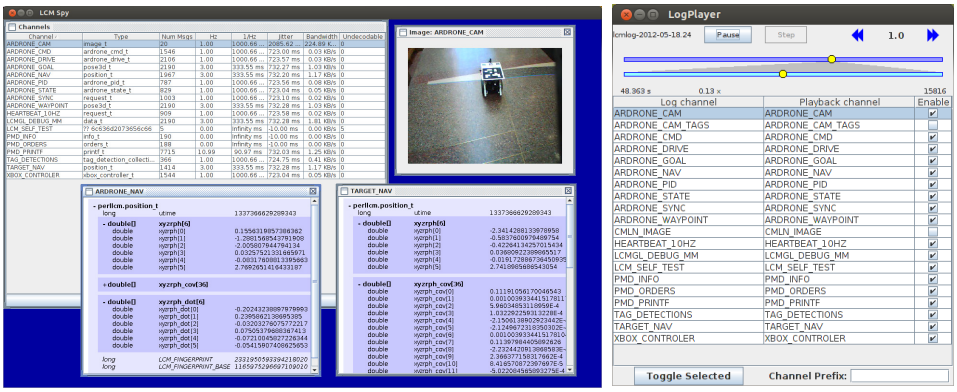

Fig. 6: The left window shows the LCM Spy tool for monitoring LCM traffic passed between processes. In view are windows displaying the real-time estimates (both mean and covariance) from the state estimator: $\mathbf{x}_v$, described by the ARDRONE_NAV message, and $\mathbf{x}_s$, described by the TARGET_NAV message. The right window shows the LCM LogPlayer for playing back recorded LCM log events.

B. Test Scenarios and Results

1) Estimation Performance Testing: The first tests of the proposed system were performed to assess the state estimation framework. We flew the UAV and observed a stationary AprilTag (representing the landing platform) in a lab environment. The testing consisted of an extended period of flying the vehicle without performing the landing maneuver with times of both viewing and not viewing the AprilTag. Ground-truth measurements of the UAV and AprilTag local-poses were recorded using a Qualisys IR Motion Capture system running in real-time at 100 Hz [23]. By simultaneously tracking the UAV and landing platform with the Motion Capture system, we quantified the performance of the EKF.

Error plots from the ground-truth tests are shown in Fig. 8. Both the EKF and motion capture system track poses of the UAV and landing platform with respect to local frames. To align and compare the EKF to ground-truth, we extracted the relative-pose from the UAV to the landing platform for each measurement method via coordinate transform operations. For the entire test shown in Fig. 8, including periods without camera observations, we recorded relative-position RMS errors of 14.4 cm, 17.6 cm, and 4.6 cm for x, y, and z respectively, and relative-orientation RMS errors under $2.5°$ for all angles. Fig. 8 shows the growth of the relative-pose covariance when AprilTag detections were lost. However, during times of steady camera observations (i.e., when the AprilTag was in continuous, uninterrupted view) the relative-pose uncertainty was significantly lower.

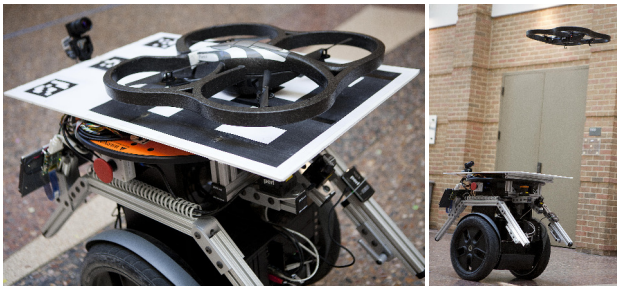

Fig. 7: (Left) The AR.Drone micro UAV sitting on the landing platform atop the Segway RMP-200 UGV. (Right) The AR.Drone during the descent phase of the landing maneuver. *Photo: Laura Rudich, Michigan Engineering Communications & Marketing*

To better understand the behavior of the estimator, we calculated the RV coefficient [25] to measure the correlation between the UAV and landing platform local-poses. The RV coefficient for random vectors is analogous to the Pearson correlation coefficient for scalar random variables. For our 6-DOF local-pose vectors, the RV coefficient is found by

$$RV(\mathbf{x}_{lv}, \mathbf{x}_{ls}) = \frac{tr(\Sigma_{\mathbf{x}_{lv}\mathbf{x}_{ls}}\Sigma_{\mathbf{x}_{ls}\mathbf{x}_{lv}})}{\sqrt{tr(\Sigma_{\mathbf{x}_{lv}}{}^2)tr(\Sigma_{\mathbf{x}_{ls}}{}^2)}} \tag{32}$$

and is plotted in Fig. 9. Since the camera observations relate both local-poses, correlation builds with each detection. Without detections, the local-poses lose correlation as the process model propagates each system (UAV and ship) according to its individual dynamics.

Also displayed in Fig. 9 is a plot of the normalized estimation error squared (NEES), used as an evaluation of filter consistency [24]. The NEES measures whether the observed filter errors (compared to ground-truth) are representative of the estimated filter covariance. We first derive the relative-pose $\mathbf{x}_{vs} \sim \mathcal{N}(\boldsymbol{\mu}_{vs}, \Sigma_{\mathbf{x}_{vs}})$ and calculate the NEES as the squared Mahalanobis distance

$$NEES[k] = (\mathbf{x}_G[k] - \boldsymbol{\mu}_{vs}[k])^\top \Sigma^{-1}_{\mathbf{x}_{vs}}[k](\mathbf{x}_G[k] - \boldsymbol{\mu}_{vs}[k]) \tag{33}$$

where $\mathbf{x}_G[k]$ denotes the ground-truth measurement of the relative-pose at the current timestep k from the motion capture system. Under the Gaussian assumption for $\mathbf{x}_{vs}$, the NEES is distributed as a chi-square random variable with six degrees of freedom, so we can plot its values during our ground-truth tests with a corresponding acceptance interval, taken in Fig. 9 to be the double-sided 95% probability region.

2) Landing Events: We then proceeded to test the operation of the full autonomous system with landing. Before including the UGV in the experimental tests, the system was tested many times with the landing platform off the UGV and placed in an elevated, stationary position. The landing platform was then attached to the UGV for testing the landing while the UGV

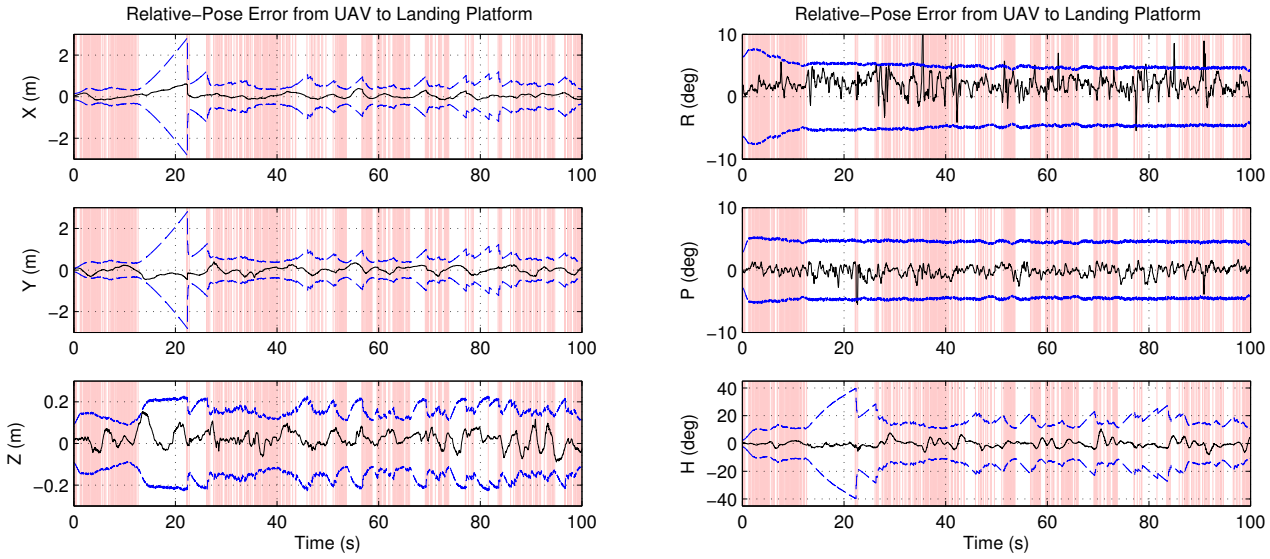

Fig. 8: Error plots from the ground-truth experiment for assessing estimation performance. Errors for the 6-DOF relative-pose from UAV to landing platform are drawn in black with associated 3-σ intervals drawn in blue. Light red lines represent timestamps where a camera observation was received. Notice the uncertainty growth during periods without camera observations; while uncertainty grows during these periods, our filtering framework allows relative-pose errors to remain small.

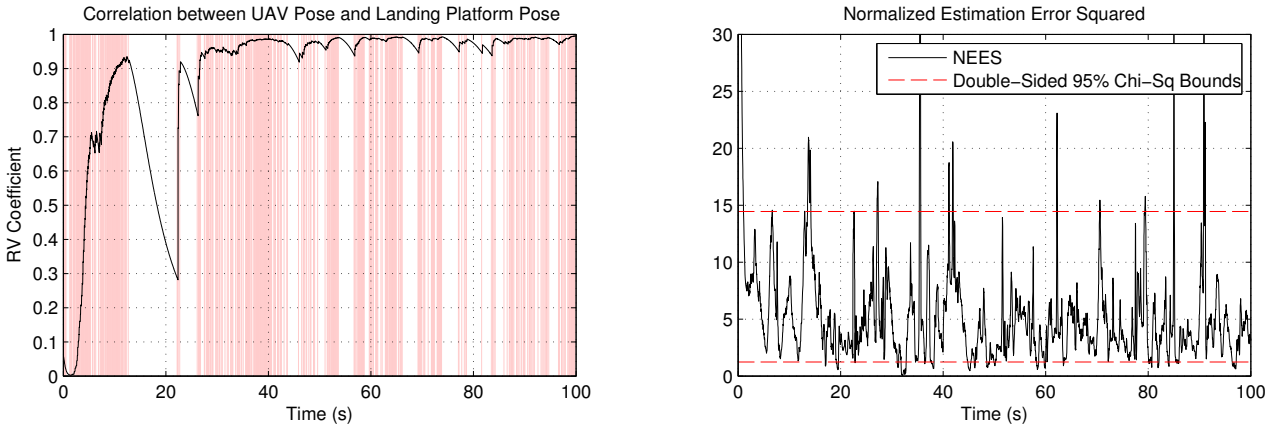

Fig. 9: (Left) The RV coefficient as a measure of correlation between the estimated UAV and landing platform local-poses during the ground-truth experiment. Light red lines represent timestamps where a camera observation was received. The poses become highly correlated when camera observations are incorporated into the state estimator, and lose correlation during periods without camera observations. (Right) Plot of normalized estimation error squared (NEES) as a measure of estimator consistency with corresponding double-sided 95% chi-square probability region [24].

traveled along a long, straight path. Photos of the experimental testbed in action are shown in Fig. 7.

Results are shown in Fig. 10 for a typical autonomous landing maneuver. The figure displays the estimated local-frame positions from the EKF for the UAV and landing platform, as well as the relative-position from UAV to landing platform with associated 3-σ bounds. In this test, the UAV took off from atop the UGV at the 7-second mark and performed a short search for the landing platform (by spinning in place), which it detected and initialized in the EKF at 9 seconds. As the UGV travelled down its path, the UAV followed from behind before beginning descent around 32 seconds and landing at the 37-second mark. Fig. 10 and Fig. 11 both show that the state estimate became more confident as the UAV approached closer to landing. This rise in confidence can be attributed to the additional detections from the small markers on the landing platform intended for fine pose control. Fig. 11 shows that the relative-pose uncertainty, measured by the sixth-root of the determinant of the covariance, shrinks to less than half its value with the additional small-marker observations. A video of the system in operation can be found at http://www.youtube.com/watch?v=7KxGFSiPLYs.

VI. Tools for Teaching, Training, and Learning

As mentioned previously, a major goal of this project and the NEEC is not only to perform research, but to educate and train young engineers in Navy-relevant autonomy. The focus on project-based education allows us to emphasize interdisciplinary collaboration and hands-on experience within our student team. A recent University of Michigan report notes

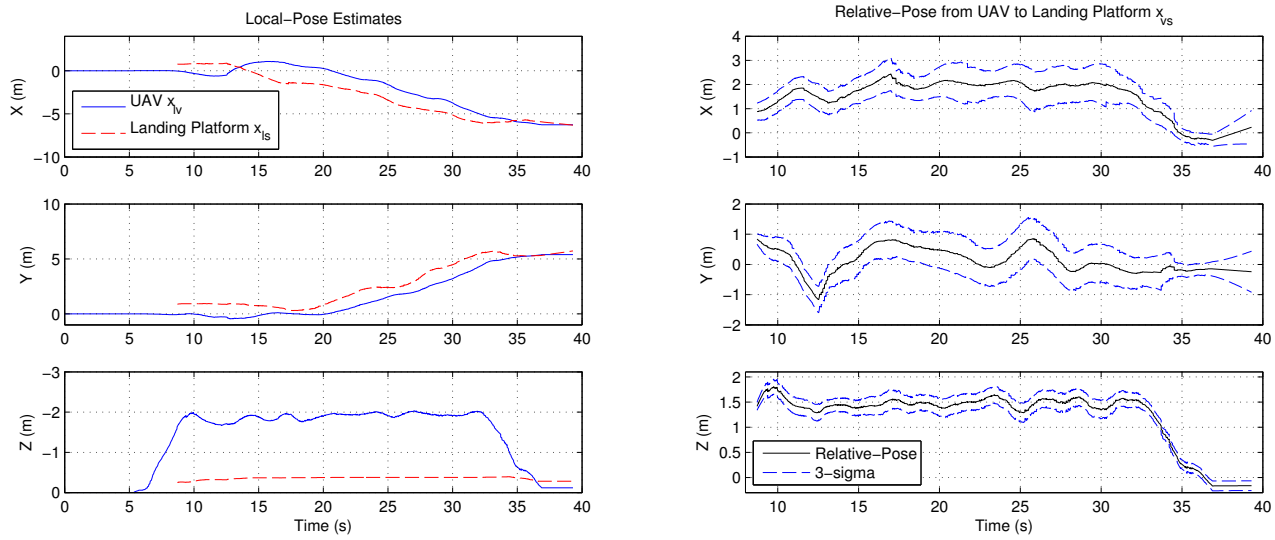

Fig. 10: Plots of relevant information for an actual in-transit landing event performed by the experimental UAV and UGV. The UAV takes off at 7 seconds and receives its first camera observation at 9 seconds, initializing the landing platform. The landing is accomplished at the 37-second mark. (Left) The UAV and landing platform positions with respect to the local NED frame, as estimated by the EKF during the landing maneuver. (Right) The relative-position from UAV to landing platform with associated 3-σ confidence bounds.

that this type of experiential learning is preferable to future employers, as it significantly enhances the ability of a student to connect classroom knowledge with professional practice [26].

Fostering a learning environment on the project begins at the team level, with a focus on experienced and older students mentoring younger students and newcomers. Brakora et. al. [26] state that successful mentoring within a team promotes year-to-year continuity of a project and can parallel the classroom curriculum. We also use the mentoring opportunity within the team to develop leadership skills and encourage collaboration. The team is composed of students across disciplines — mechanical engineering, electrical engineering, and computer science — so collaboration also provides a method for teaming individual specializations and skillsets to address a problem.

This collaborative environment favors the modular software architecture that we designed for the proposed system. Team members can work in groups on a specific component of the system software and use common LCM type definitions for bridging communication between modules developed by different groups. In this way, LCM not only serves as a nice interface between processes, but provides transparency of the entire software architecture so that team members can better understand how the system operates as a whole.

VII. Conclusions

We presented a UAV system for autonomously landing on the deck of a ship at sea. With the aid of only onboard sensors and visual fiducial markers on the desired landing platform, the system can autonomously track and perform landing maneuvers onto a transiting ship. Our state estimator provides robustness that is not offered by current visual-servoing based methods. The addition of delayed-states in our extended Kalman filter formulation allows us to circumvent issues with image processing latency — increasing the reliability of the system. It is essential for military UAVs to eliminate their reliance on external signals such as GPS and UAV-to-ship communications, and our system allows UAVs to safely return without such added infrastructure.

To simplify the landing platform detection process, we strategically placed AprilTag markers around the platform to guide our autonomous UAV on a safe descent. In future iterations, we expect that AprilTag markers will be replaced by arbitrary images that can be learned by the image processing unit, such as in Ferns classification [27].

Finally, of paramount importance is the fact that this project was completed by a team of voluntary NEEC students. By exposing undergraduate students to real-world Navy problems at an early stage in their engineering career, we are preparing them to tackle even more difficult challenges in future naval applications.

VIII. Acknowledgments

The authors would like to thank the NEEC team of students for their hard work in supporting the project: Jeffrey Chang, Nicholas Fredricks, Michelle Howard, Timothy Jones, Noah Katzman, and Angelo Wong. We would also like to acknowledge our Navy POCs: Frank Ferrese at NAVSSES, Paul Young at NSWC-DD, and Roger Anderson at NSWC-PCD.

References

[1] "The Navy Unmanned Undersea Vehicle (UUV) Master Plan," U.S. Navy, Tech. Rep., November 2004.

[2] "The Navy Unmanned Surface Vehicle (USV) Master Plan," U.S. Navy, Tech. Rep., July 2007.

[3] J. Paur, "Can killer drones land on carriers like human top guns?" November 2009. [Online]. Available: http://www.wired.com/dangerroom/2009/11/can-killer-drones-land-on-carriers-like-human-top-guns/

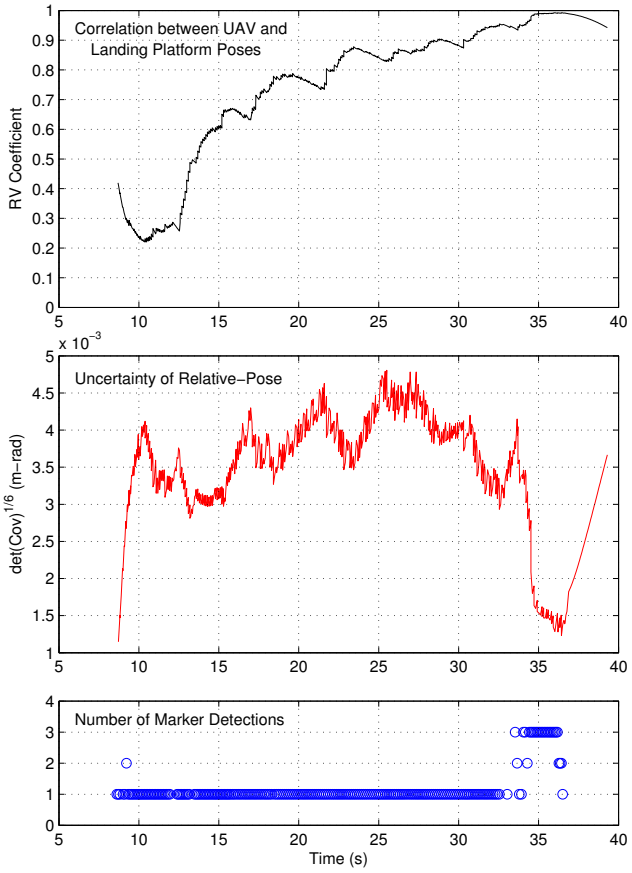

Fig. 11: (Top) The RV coefficient as a measure of correlation between the UAV and landing platform local-poses. Correlation builds as camera observations to the platform are incorporated by the state estimator. Landing occurs at the 37-second mark. (Middle) The uncertainty of the relative-pose from UAV to landing platform, taken as the sixth-root of the determinant of the covariance of the relative-pose [19]. Detections of the small markers cause the uncertainty to shrink quickly immediately before the landing. (Bottom) The number of marker detections during the landing event. As the UAV approaches close to the platform right before landing, the three small markers come into view, causing an increased number of observations beneficial for fine pose control.

[4] L. Coutard, F. Chaumette, and J.-M. Pflimlin, "Automatic landing on aircraft carrier by visual servoing," in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), September 2011, pp. 2843–2848.

[5] Sierra Nevada Corporation, "Firescout lands at sea using SNC's UCARS-V2!" January 2006. [Online]. Available: http://www.sncorp.com/pdfs/SNC_news/Firescout%20Press%20Release.pdf

[6] J. Kelly, "Unmanned X-47B readies for final touchdown," July 2013. [Online]. Available: http://navylive.dodlive.mil/2013/07/09/unmanned-x-47b-readies-for-final-touchdown/

[7] S. Ackerman, "Exclusive pics: The Navy's unmanned, autonomous UFO," July 2012. [Online]. Available: http://www.wired.com/dangerroom/2012/07/x47b

[8] S.-R. Oh, K. Pathak, S. K. Agrawal, H. R. Pota, and M. Garratt, "Autonomous helicopter landing on a moving platform using a tether," in Robotics and Automation (ICRA), 2005 IEEE International Conference on, 2005, pp. 3960–3965.

[9] M. Garratt, H. Pota, A. Lambert, S. Eckersley-Maslin, and C. Farabet, "Visual tracking and LIDAR relative positioning for automated launch and recovery of an unmanned rotorcraft from ships at sea," Naval Engineers Journal (NEJ), vol. 122, no. 2, pp. 99–110, 2009.

[10] B. Herisse, T. Hamel, R. Mahony, and F.-X. Russotto, "Landing a VTOL unmanned aerial vehicle on a moving platform using optical flow," IEEE Transaction on Robotics, vol. 28, no. 1, pp. 77–89, February 2012.

[11] S. Arora, S. Jain, S. Scherer, S. Nuske, L. Chamberlain, and S. Singh, "Infrastructure-free shipdeck tracking and autonomous landing," in Robotics and Automation (ICRA), 2013 IEEE International Conference on, 2013, pp. 323–330.

[12] T. S. Richardson, C. G. Jones, A. Likhoded, E. Sparks, A. Jordan, I. Cowling, and S. Wilcox, "Automated vision-based recovery of a rotary wing unmanned aerial vehicle onto a moving platform," Journal of Field Robotics, vol. 30, no. 5, pp. 667–684, 2013.

[13] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," Autonomous Robot Vehicles, vol. 1, pp. 167–193, 1990.

[14] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), May 2011, pp. 3400–3407.

[15] R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, 2nd ed. Cambridge University Press, 2003.

[16] J. J. Leonard and R. Rikoski, "Incorporation of delayed decision making into stochastic mapping," in Experimental Robotics VII, ser. Lecture Notes in Control and Information Sciences, D. Rus and S. Singh, Eds. Springer-Verlag, 2001.

[17] R. F. Stengel, Optimal Control and Estimation. Dover, 1994.

[18] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-92-01, January 1992.

[19] H. Carrillo, I. Reid, and J. Castellanos, "On the comparison of uncertainty criteria for active SLAM," in Robotics and Automation (ICRA), 2012 IEEE International Conference on, 2012, pp. 2080–2087.

[20] A. Huang, E. Olson, and D. Moore, "LCM: Lightweight communications and marshalling," in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), October 2010, pp. 4057–4062.

[21] Parrot SA., "AR.Drone." [Online]. Available: http://ardrone.parrot.com

[22] Segway Inc., "Segway RMP-200." [Online]. Available: http://rmp.segway.com

[23] Qualisys, "Qualisys ProReflex MCU 1000." [Online]. Available: http://www.qualisys.com

[24] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, Estimation with Applications to Tracking and Navigation: Theory, Algorithms, and Software. Wiley, 2001.

[25] P. Robert and Y. Escoufier, "A unifying tool for linear multivariate statistical methods: The RV-Coefficient," Journal of the Royal Statistical Society. Series C (Applied Statistics), vol. 25, no. 3, pp. 257–265, 1976.

[26] J. Brakora, B. Gilchrist, J. Holloway, N. Renno, S. Skerlos, T. Teory, P. Washabaugh, and D. Weinert, "Integrating real-world experience into a college curriculum using a multidisciplinary design minor," in 116th Annual American Society of Engineering Education Conference & Exposition, no. AC 2009-2282, June 2009.

[27] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, pp. 448–461, 2010.