Improving the Reliability of Pick-and-Place With Aerial Vehicles Through Fault-Tolerant Software and a Custom Magnetic End-Effector

Gowtham Garimella, Matthew Sheckells, Soowon Kim, Gabriel Baraban, and Marin Kobilarov

***Abstract*—Aerial manipulation is an emerging field in robotics with various potential applications such as transport and delivery, agriculture, and, infrastructure inspection. To deploy aerial vehicles in the real world, the safety and reliability of these systems is paramount. Motivated by the need for safety and reliability, this work proposes a software framework that has built-in robustness to algorithmic failures and hardware faults. The framework allows users to build complex applications while reasoning about faults that can happen at different stages of an aerial manipulation task and specifying fallback actions to return to normal operating mode. The aerial manipulator is further endowed with a magnetic gripper that can handle positional errors arising from perception and control uncertainties. We also introduce a bias estimator for measuring the contact forces and sensor bias. We demonstrate how the estimator can be used to detect either completion or failures across several tasks. We demonstrate the reliability of the proposed framework on two tasks: package sorting task (e.g. as might be used in a distribution center) and sensor placement task (for infrastructure inspection). We show different failure modes that can occur and how our aerial manipulation system recovers from them.**

***Index Terms*—Planning, scheduling and coordination, software architecture for robotic and automation, factory automation.**

I. Introduction

VERTICAL take-off and landing (VTOL) vehicles such as quadrotors have gained recent attention due to their agility and ability to navigate in remote and cluttered environments. Current research suggests that VTOL vehicles attached with manipulators, known as aerial manipulators, are attractive for numerous applications, including package transportation [1], collaborative load transportation [2], collaborative construction, vision based target interception [3], and structural maintenance applications [4], [5].

Previous research efforts on aerial manipulation focused on building novel control techniques and hardware to push the boundary of applications for manipulators [6]. Aerial manipulators are usually applied in situations requiring a high level of safety and reliability. For example, package transportation requires that the aerial manipulator detect the packages are safely picked up and sensor placement tasks require that the sensors are placed firmly on the target.

We propose a two-fold approach to improve the reliability of the aerial manipulators: on the software side, a fault tolerant state machine framework which has estimators and monitors that can in real time detect contact forces and faults, and, on the hardware side, a novel magnetic gripper that tolerates end-effector error up to 2 cm while grasping. The result is a reliable aerial manipulation system that is demonstrated on a pick-and-place scenario and remote sensor payload placement task. We hope that the aerial manipulation software will provide a base for researchers to develop custom aerial manipulation applications with improved reliability.

A. Related Work

Past research has focused on developing novel control algorithms and manipulators for aerial manipulation tasks. Manipulators range from rigid arms [7], [8] to compliant arms [9], [10], and more recently novel compliant bimanual aerial manipulators have also been proposed in [11], [12]. In this work we use a simple 2DOF arm which is sufficient for the tasks considered in this work. An omnidirectional aerial manipulator that can improve the stability of the aerial manipulator has been proposed by Bodie *et al*, [13].

Several novel control techniques for aerial manipulation have been proposed in [6], [14]–[16]. Benchmarks for the control performance of aerial manipulators have been proposed in [17]. In this work we use control algorithms that can be applied to off the shelf quadrotors that can be retrofitted with a manipulator. We focus on improving the reliability of the system without using a sophisticated controller.

A few fully integrated applications for aerial manipulation have been proposed in recent years. An aerial manipulation

Manuscript received March 1, 2021; accepted June 12, 2021. Date of publication June 30, 2021; date of current version August 13, 2021. This letter was recommended for publication by Associate Editor T. Liu and Editor J. Yi upon evaluation of the reviewers' comments. This work was supported by NSF award: 1925189. *(Gowtham Garimella and Matthew Sheckells contributed equally to this work.) (Corresponding author: Marin Kobilarov.)*

Gowtham Garimella was with the Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD 21218, U.S., and now with Zoox Inc., Foster City, CA 94404, U.S. (e-mail: marin@jhu.edu).

Matthew Sheckells was with the Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218, U.S., and now with the SpaceX, Hawthorne, CA 90250, U.S. (e-mail: garimella.gowtham74@gmail.com).

Soowon Kim was with the Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD 21218, U.S., and now with the Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, U.S. (e-mail: mshecke1@jhu.edu).

Gabriel Baraban and Marin Kobilarov are with the Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: soowk311@gmail.com; gbaraba1@jhu.edu).

Digital Object Identifier 10.1109/LRA.2021.3093864

system for moving metallic discs and sheets is proposed by [18], [19]. The system developed by Gawel *et al.* [18] used an electro-permanent gripper that can turn on and off the magnetic effect by reversing an electric current. In contrast, our work proposes a permanent magnetic gripper solution that can turn on and off by changing the polarity of the magnets using a mechanical servo. This type of gripper does not use energy to hold the object and only requires momentary energy to release objects. Lee *et al.* proposed a collaborative framework for moving an unknown object in an unknown obstacle ridden environment [20]. Kim *et al.* developed an aerial manipulation system for lab automation using a parallel manipulator [21]. Orsag *et al.* suggested a benchmark for different aerial grasping applications [22]. Our work performs two similar benchmark applications: grasping objects from a table and placing them in slots on a shelf, and placing an adhesive sensor payload on a remote surface.

An open source software package for aerial manipulation has been proposed by Perez *et al*, [23]. The software framework proposed in this work focuses on building aerial manipulation applications whereas the above package provides several hardware options and controllers for manipulators.

B. Organization

The rest of the article is organized as follows. The basic software framework for building complex aerial manipulation experiments and the controllers needed are described in Section II, III. We define the types of faults that can be detected, and the online estimators and monitors needed to make the software framework fault tolerant in Section IV. The hardware modifications to make the system more reliable are explained in Section V. The aerial manipulation system is evaluated on two tasks in Section VI and the conclusions are presented in Section VII.

II. Software Framework

At the core of the aerial manipulation system lies a software framework with several important capabilities. The software framework has been designed to: combine modular behaviors into complex state machines to perform complicated tasks; enable robustness to sensor, controller, and hardware failure, through introspection and fail-safe actions; provide a simulation environment for testing the system before actual deployment; automate tests for controllers and logic systems, independent of their hardware implementation; serve as an open-source system for developing complex aerial autonomy applications. It tightly integrates high-level control strategies for both quadrotors and manipulators with an existing finite state machine library to provide robustness to controller and hardware failures during the task. The framework consists of several modular features, such as hardware drivers, controllers, and visual trackers. Figure 2 illustrates the interaction between different components of the robot system.

Fig. 1. Proposed aerial manipulation system picking (top) and placing (bottom) a package.

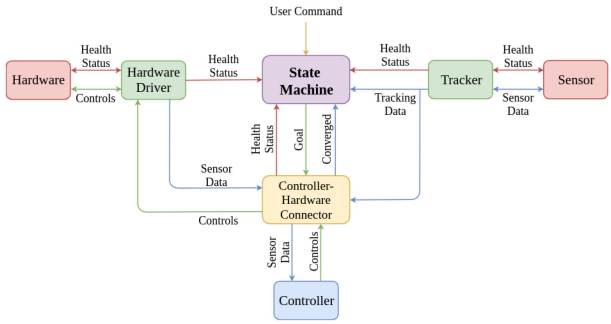

Fig. 2. Illustration of the interaction between the various software components of the developed framework.

III. Aerial Manipulator Control

We now describe two of the trajectory tracking controllers implemented on our aerial manipulation system: an acceleration-based controller that relies on roll-pitch-yaw-thrust commands and an MPC controller.

A. Acceleration-Based Control

Define the state of the quadrotor as $x = (p, R, v, \omega)$, where $p \in \mathbb{R}^3$ is the position, $R \in SO(3)$ is the rotation matrix, $v \in \mathbb{R}^3$ is the velocity, and $\omega \in \mathbb{R}^3$ is the angular velocity. The autopilot takes as input the desired roll ϕ_d, desired pitch θ_d, desired yaw rate $\dot{\psi}_d$ and a thrust command $u_t \in \mathbb{R}$. It internally runs a feedback loop that controls the rotor velocities to achieve these high-level commands. The aim of the controller is to accurately track a desired reference trajectory in terms of position, velocity, and yaw, where the reference is specified as a smooth trajectory in quadrotor position $p_r \in \mathbb{R}^3$ and quadrotor yaw ψ_r. To achieve this task, we design a controller that computes the desired acceleration $a_d \in \mathbb{R}^3$ based on the error in position $e_p = p_r - p$ and error in velocity $e_v = \dot{p}_r - v$ as

$$a_d = K_p e_p + K_d e_v + a_r, \tag{1}$$

where $K_p, K_d \in \mathbb{R}^{3\times3}$ are positive diagonal matrices that act as proportional and derivative gains and $a_r = \ddot{p}_r$ is the feedforward acceleration based on the reference trajectory.

Next, we compute the roll, pitch, and thrust commands that achieve the desired acceleration a_d. The rotors on the quadrotor are aligned with the body z-axis, which implies the quadrotor can only apply acceleration along this axis. The net acceleration produced by the quadrotor is given by

$$a = R_Z(\psi)R_Y(\theta)R_X(\phi)e_3 u_t - g, \tag{2}$$

where ψ, θ, and ϕ represent a ZYX Euler parametrization of R, $R_{(\cdot)}$ represents rotation about z, y, and x-axes, $g = [0, 0, -9.81]$ is the gravity vector and $e_3 = [0, 0, 1]^T$ is the body z-axis. Mass does not enter the equation as u_t is a commanded body z-axis acceleration rather than a true thrust force. We solve for the autopilot inputs ϕ, θ, and u_t by setting a as a_d. The desired thrust command is given by

$$u_t = \|a_d + g\|. \tag{3}$$

To find the desired roll and pitch, we define the normalized acceleration vector as $\bar{a}_d = (a_d + g)/u_t$. The desired roll and pitch are then given by

$$\phi_d = \arcsin(\bar{a}_d^\top e_1 \sin\psi - \bar{a}_d^\top e_2 \cos\psi), \tag{4}$$

$$\theta_d = \arctan\left(\frac{\cos\phi(\bar{a}_d^\top e_1 \cos\psi + \bar{a}_d^\top e_2 \sin\psi)}{\cos\phi\ \bar{a}_d^\top e_3}\right). \tag{5}$$

The maneuver during the tasks is limited to avoid any singularities during the conversion. The commanded yaw rate is proportional to the error between the current yaw and desired yaw obtained from the reference trajectory as

$$\dot{\psi}_d = k_\psi(\psi - \psi_r) + \dot{\psi}_r, \tag{6}$$

with $k_\psi \in \mathbb{R} > 0$.

Previous work proves stability for a similar class of trajectory tracking controllers that use Proportional–Integral–Derivative (PID) controller to compute a desired force and an inner-loop attitude controller to achieve the desired force direction [24]. We assume the arm is attached to the Center of Gravity (CoG) of the quadrotor and arm dynamics are relatively slow compared to the quadrotor dynamics. Under these assumptions, the arm is assumed to be a kinematic system and is controlled independently of the quadrotor without affecting the overall stability of the system.

B. Model Predictive Controller

The Model Predictive Controller (MPC) computes the thrust and desired rotation matrix for the quadrotor by solving a trajectory optimization problem. The trajectory optimization problem at a high level can be written as

$$u^*_{1:N} = \underset{u_{1:N}}{\operatorname{argmin}} (x_N - \bar{x}_N)^\top Q_N (x_N - \bar{x}_N) + \sum_{i=0}^{N-1} (x_i - \bar{x}_i)^\top Q (x_i - \bar{x}_i) + (u_i - \bar{u}_i)^\top R (u_i - \bar{u}_i), \tag{7}$$

$$x_{i+1} = f(x_i, u_i), \tag{8}$$

where x_i is the state of the aerial manipulator, $\bar{x}_i$ is the reference state, u_i is the control, and $\bar{u}_i$ is the reference control, and N is the number of trajectory steps.

The optimization minimizes the cost over a predicted trajectory for the aerial manipulator using a sequence of control inputs subject to the dynamics of the system and other application based constraints. The advantage of MPC over traditional controller is that it can handle constraints such as obstacles and the interaction between the manipulator and the rotor base.

We support using the full aerial manipulator dynamics as explained in [25]. The inputs to the MPC are the joint torques and the rotor thrusts which are mapped to the body torques and body thrust. Since we use an off the shelf quadrotor, we can only input the desired Euler angles of the rotor base and the joint angles. These are picked from the reference states of the optimal trajectory obtained through MPC.

A simplified second order model of the quadrotor rotational dynamics as shown in [26] can also be used for the MPC. This system assumes the interactions between the quadrotor and the arm as external disturbances and tries to apply controls to compensate for these external disturbances and follow the optimal trajectory. This approach allows for independent control of quadrotor and the manipulator. Although not as efficient as the above apprroach, it simplifies the MPC problem and the optimization can be done at a much higher rate.

The system dynamics used in MPC consists of unknown system parameters that need to be estimated. For example for the simplified MPC, we need to estimate the external disturbances modeled as accelerations on the quadrotor base. We use online estimators described in Section IV to estimate and update the system parameters continuously. We also provide a way to manually control the system or perturb the system automatically around the point of operation to estimate the model parameters.

C. Reference Trajectory Generation

Two strategies are used to generate reference trajectories for navigation and manipulation purposes.

1) Navigation: When navigating to a waypoint or approaching a target object, we use a polynomial reference trajectory of degree 9 along each individual axis to ensure the reference derivatives are smooth up to fourth order. The coefficients of the polynomial are found by solving a linear system defined by the boundary conditions of the trajectory, where the initial position and yaw are given by sensors and final position and yaw by the user. The rest of the derivatives of the position at the boundaries are set to zero so that the trajectory starts and ends at rest.

2) Grasping Strategy: Close to the object in the final stage of the picking procedure, we track a trajectory that is constant in the plane parallel to the object, but sinusoidal perpendicular to the object, resulting in a periodic "poking" motion. This behavior pushes the end-effector towards the object with the intent of making contact during the first half cycle of the motion, but pulls the end-effector back away from the object if it is misaligned while poking. By pulling away, the robot has the opportunity to correct its attitude and relative position without colliding with the object before the next poking cycle begins.

TABLE I
FAULT MONITORS

Monitor	Description
Quadrotor health	Check the battery health and readiness of quadrotor.
Manipulator health	Monitor faults arising from servo such as overload, joint limits etc
Contact force monitor	Detect external forces on aerial manipulator
Command monitor	Detect unsafe commands being sent to quadrotor and manipulator.

IV. FAULT TOLERANCE

Reliable aerial manipulation tasks requires the monitoring and detection of faults online, in order to make corrective actions in real-time. In this work we deal with faults arising from algorithmic failure, triggered e.g. when the autonomous navigation software failed to reach its target, and hardware failures triggered e.g. when the manipulator detects high contact forces. Not all possible faults are recoverable. For example, the loss of a rotor, a mechanical failure of the manipulator, or, a seg-fault of user-specified state-machine are out of scope for this work. In such extreme cases, the fallback behavior is to employ a basic strategy such as immediate landing or hovering in place if applicable.

The fault monitors used in this work are described in Table I. These monitors are integrated into state machine framework to either abort current action or perform recovery action. For instance, if the vehicle battery is below a threshold, we can land if it is safe to do so. Similarly if the arm is subjected to a large external contact force (measured through an estimator), we can switch off the arm power. We also incorporated a command monitor which detects if the commands being sent to the vehicle and its manipulator are unsafe. For example, we can detect if the desired position and yaw is outside a geo-fence or will lead to collision. We can also detect if the commands to the arm will cause self collisions. The commands can either be clipped or a fault raised and the action can be aborted.

In addition to the generic monitors, each individual algorithm can monitor its status and detect failures to reach target and abort or retry. In this work, we use monitors to, detect if we are able to pick a package, drop a package, place a sensor on wall, measure the progress of each action, and, retry after a time-out.

The online monitors rely on estimating a number of vehicle state parameters online. We next describe the estimators needed to build the generic monitors.

A. Thrust Gain Estimation

The autopilot takes as input a normalized thrust command between 0 and 100, where a non-constant scale factor transforms the normalized value to a metric unit of thrust force. The scale factor, called the thrust gain, is constantly changing as it depends on the battery voltage and mass of the quadrotor. Since the input to the MPC and the controller is the quadrotor's thrust force, a thrust gain estimator computes the mapping between the thrust command and the actual thrust force. We combine the mass into the thrust gain to directly map the normalized input to gravity compensated acceleration of the quadrotor. The commanded thrust $u \in \mathbb{R}$ maps to a corresponding global acceleration $a \in \mathbb{R}^3$ of the quadrotor as

$$a = k_t R e_3 u + g \tag{9}$$

where $k_t \in \mathbb{R}$ is the thrust gain, the orientation of the body is denoted by the rotation matrix R, and the thrust vector is assumed to be pointed towards the body-z direction, i.e. $e_3 = [0, 0, 1]$.

The thrust gain can be obtained from the measured body acceleration $a_b \in \mathbb{R}^3$ and gravity vector as

$$k_t = \frac{1}{u} e_3^T (a_b - R^\top g) \tag{10}$$

These measurements can be obtained from the Inertial Measurement Unit (IMU) on the quadrotor. The noise in the IMU measurements is accounted for by using an exponential filter

$$\bar{k}_{t_{i+1}} = \bar{k}_{t_i} + \lambda (k_{t_i} - \bar{k}_{t_i}), \tag{11}$$

where $\bar{k}_{t_i}$ is the filtered thrust gain estimate at time index i. By increasing the scaling parameter λ from 0 and 1, the thrust gain can be adjusted to change more aggressively, which leads to the quadrotor changing thrust faster to compensate for a change in mass. Figure 3 shows the thrust gain estimated for the quadrotor during a pick-and-place application. The positive jumps in the gain coincide with a package being dropped and a negative jump coincides with a package being picked up. The thrust gain exhibits an overall downward trend as the battery voltage drops over time.

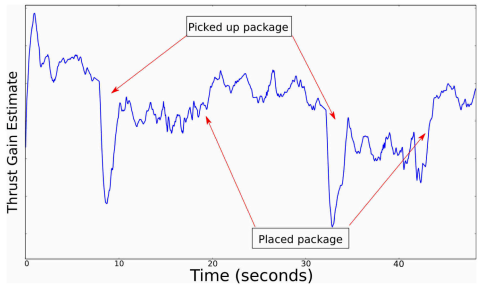

Fig. 3. Estimate of thrust gain k_t computed from IMU data and expected acceleration during pick-and-place task.

B. Euler Angle Bias Estimation

We found a small difference of approximately 0.5° between the roll and pitch reported by the IMU and the angles obtained by inverting the fused body acceleration reported by the IMU a_b. The roll and pitch angles corresponding to fused body acceleration ϕ_{acc}, θ_{acc} are obtained using (4), (5) where desired a_d is replaced by the rotated body acceleration reported by the IMU (a_i), that is

$$a_i = R_Y(\theta) R_X(\phi) a_b, \tag{12}$$

and the normalized acceleration vector is given by $\bar{a}_i = (a_i + g)/\|a_i + g\|$. To track the reference trajectory, we need to track Euler angles that are consistent with the body acceleration. Hence, we add the difference between the angles $\delta_\phi, \delta_\theta$ to the

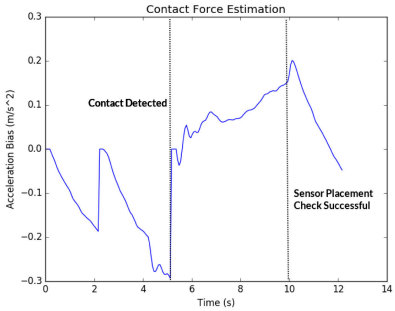

Fig. 4. The acceleration bias estimate during a sensor placement trial. The system uses a threshold on the contact force to determine when the payload has been pushed against the target surface. Once firmly pressed against the surface and before releasing the payload, the robot pulls on the payload to ensure that it successfully adhered to the surface.

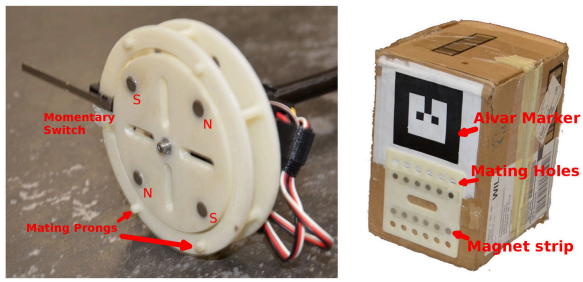

Fig. 5. The magnetic gripper (left) and a sample package (right) used in our aerial manipulation experiments. The package is instrumented with an AR marker to facilitate tracking and a magnetic mating joint so it can attach to the gripper.

commanded roll and pitch before sending them to the autopilot, where

$$\delta_\phi = \phi - \phi_{acc},\ \ \delta_\theta = \theta - \theta_{acc}. \tag{13}$$

C. Contact Force Bias Estimation

For applications that involve computing a contact force with the environment, such as placing a sensor payload on a surface with a specified amount of force, the system must estimate the acceleration bias induced by these contact forces. Unfortunately, due to the normalization of the thrust command as described above, we cannot estimate force explicitly. Instead, we estimate the acceleration bias using the difference between the IMU reading and the expected thrust acceleration as

$$a_{bias} = Ra_b - k_t Re_3 u - g. \tag{14}$$

For applications like sensor placement, we use the local x-coordinate of the bias vector for force estimation. When the gripper is pressing against a wall, this quantity is negative, while a force pulling on the gripper results in a positive estimate. Just as with the other estimators in this section, the values calculated by this formula are smoothed with an exponential filter to reduce noise. Figure 4 shows an example contact force estimation trajectory from a sensor placement trial. The acceleration bias, when there is no external contact forces, is constant as shown in the first section of the Figure 4. When there are external forces, the acceleration bias keeps going up in proportion to the external force being applied. Thus the difference in bias can be used as a measure of the contact force applied. A newer approach using an adaptive estimator to measure contact force in Newtons has also been proposed in [27]. We found that the simpler approach to estimate the contact force bias is sufficient for the applications in this work.

V. Hardware

A. Commercial Off-the-Shelf quadrotor

The aerial manipulation system uses a modified DJI Matrice quadrotor as the base. The quadrotor is equipped with a PointGrey Flea3 camera and an Intel NUCi5 computer, which communicates with the Matrice flight controller.

B. Manipulator

1) Custom 2-DoF Arm: Several previous works, like [28] and [7], develop arms specifically for aerial manipulation, but they typically only grasp objects directly below the robot and cannot reach outside the envelope of the quadrotor. In this work, a light-weight 2-DoF manipulator is used for picking objects outside the envelope of the quadrotor. Dynamixel servos control the manipulator joints which are connected by carbon fiber tubes. The manipulator end-effector is steered using a Cartesian position controller which commands joint velocities to achieve a desired end-effector position. As the arm is underactuated, the pose of the end effector can only be specified using two translational coordinates.

2) Magnetic Gripper: The arm uses a custom gripper to pick and place objects. As the position accuracy of the quadrotor is limited to around 2 centimeters, the gripper should be able to pick the object without requiring a high degree of precision. The gripper also needs to be able to pick objects of different sizes and shapes. Existing open-source grippers, such as the Yale Open-Hand [29], are too heavy and do not fit the requirements specified above. Our custom gripper shown in Figure 5 is composed of four magnets with alternating polarity embedded into a wheel attached to a servo. The magnets are attracted to a mating joint that is attached to the target object. The mating joint has a pattern of magnets to provide several mounting points to have a higher tolerance ($\leq$3 cm) of the position error.

Once an object is attached to the gripper, it can be released by rotating the magnet wheel 90° which flips the polarity of the magnets and repels the object. The gripper uses a momentary switch to detect whether it has attached to a mating joint, allowing the onboard computer to know when it has successfully picked up an object.

VI. Experiments

We evaluate the aerial manipulation system on two applications: Package sorting, Sensor placement. In both applications, we demonstrate fault recovery capability of the aerial manipulation system.

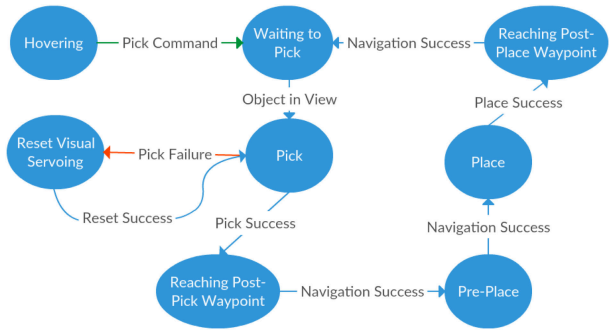

Fig. 6. Part of the state machine for picking and placing a package. The recovery actions are red and user actions are green. The user can also abort from any other state back to hovering if manual intervention is desired.

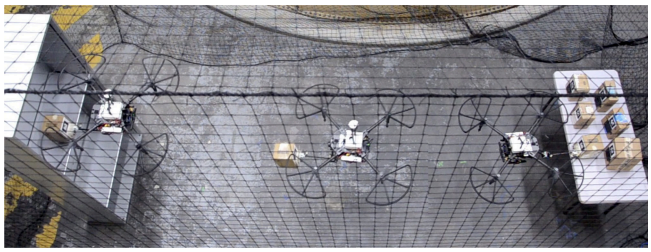

Fig. 7. An overhead view of the pick-and-place procedure.

A. *Package sorting Application*

The software framework developed in §II is used to develop a package sorting application leveraging the aerial manipulation platform described in §V.

The goal of the application is to sort packages from a packaging area (table) and transport them to corresponding storage area (shelf) to demonstrate the system's reliable aerial grasping and object insertion capabilities.

The packages are tagged with AR markers [30] and have an attached mating joint that connects to the gripper described in §V-B2. Each package has a corresponding destination marker ID where the object is placed. Figure 7 shows a timeline of the quadrotor picking and transporting packages to their corresponding storage spaces. The packages have masses between 120g and 170g. The mass of the package is limited by the arm capacity (200g) and the quadrotor payload capacity (500g).

Figure 6 shows a simplified illustration of the finite state machine for the pick place application.

The aerial manipulation system starts the package sorting from "Waiting to Pick" state, automatically detects the closest available package in the workspace, picks up the package, determines the storage location based on the marker ID of the object picked up, uses visual servoing using on-board camera to navigate to a marked shelf, places the package on the shelf, and returns to a start position with the packages in view. This process is repeated indefinitely assuming new packages appear continuously in the packaging area.

We use all the generic monitors mentioned in Table I during the package sorting application. We classify the faults into two categories: Recoverable, Unrecoverable. For recoverable faults, the software framework automatically retries algorithms and the experiment is not interrupted. Unrecoverable faults are cases where user interruption is needed but the software framework is still safe. We did not encounter any faults that could not be handled by the software framework among all the experiments done in this work.

TABLE II
RECOVERABLE FAULTS AND CORRECTING ACTIONS

Recoverable Faults	**Action**
Package not in Camera view	Retract to a previous known location and retry
Package not at a known accessible location	Abort pick
Package too heavy to pickup based on thrust gain	Abort picking package
Gripper failed to pick package based on contact sensor	Retract and retry
Lost package during placement	Abort place and go to pick.

TABLE III
PICK-AND-PLACE TASK STATISTICS

Unrecoverable faults	4
Recoverable faults	21
End-to-end Package delivery rate	85/101

The recoverable faults handled by the software framework are described in Table II. In the context of experiment recovery implies the mission can go forward even if it will not be able to complete transporting a particular package.

The unrecoverable faults encountered during the experiment are: Losing motion capture control and Camera driver failure. In these cases the aerial manipulator would hover in place using low-level quadrotor controller provided by DJI. Manual operator would then take over the control of quadrotor and bring it to a recoverable state before continuing the mission.

We quantified the ability of the quadrotor to perform a successful pick operation over 101 trials of picking and placing. Table III shows the stats from the pick place trials. There were 4 unrecoverable faults that required manual intervention. There were around 21 recoverable faults out of which we lost a package 12 times, but the mission continued autonomously. Overall we were able to transport 85 packages end to end with 4 manual interventions. Without a recovery process, the system would have resulted in 25 manual interventions i.e 21 recoverable and 4 unrecoverable. Our system managed to bring down the manual interventions to only 4 and even then, ensured the UAV is safe until manual intervention. The package transportation rate also improved due to the recovery process by successfully transporting a package despite a recoverable fault in 9 out of 21 times. The rate of recoverable and unrecoverable faults can be further minimized with better software and hardware and will be the focus of future work.

The performance of the controllers is shown in table IV. The acceleration-based controller is used for these trials since it was easier to tune and performed slightly better than MPC at the picking task. Figure 8 compares the mean absolute errors along translational positions, velocities, and yaw angle for each controller. Both the MPC controller and acceleration-based

TABLE IV
CONTROLLER STATISTICS

Min Pick Time	6.5 seconds
Mean Pick Time	11.5 seconds
Max Pick Time	25 seconds
Mean Absolute Error x	2.1cm
Mean Absolute Error y	2.5cm
Mean Absolute Error z	1 cm
Mean Absolute Error ψ	0.03 rad

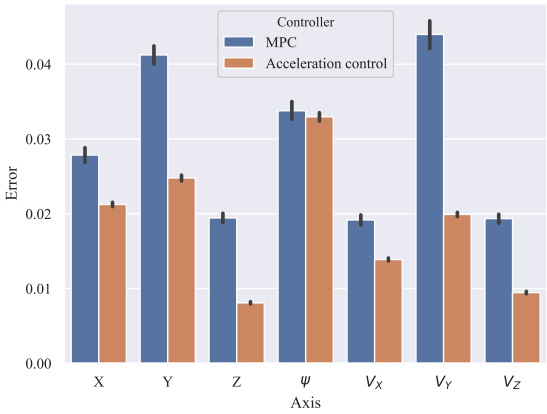

Fig. 8. Mean absolute errors along x, y, z (meters), and yaw ψ axes (radians) and translational velocities (meters/second) for MPC and acceleration-based controller. The black lines show the 95% confidence interval obtained using bootstrapping.

controller performed well during trajectory tracking, but the acceleration-based controller with more extensive gain tuning produced slightly better results.

Figure 7 shows a timelapse of the pick-and-place task, where the quadrotor picks up a package from the table and places it in a shelf. The media attachments associated with this work demonstrate the complete pick-and-place task where the quadrotor sorts multiple packages into the top and bottom shelves without any manual interruptions.

B. Remote Sensor Payload Placement

Re-using many of the same behaviors from §VI-A, we leverage our software framework to develop a sensor placement task, where the robot autonomously places a camera on a remote surface. A remote operator specifies a Region of Interest (ROI) on an onboard camera image, and the aerial manipulator visually servos to the ROI and deploys the payload safely to the chosen location, using an external force estimater to identify contact with the surface.

The sensor placement software is composed of 3 parts: an ROI tracker, an external force estimator (described in IV-C), and a state machine.

1) ROI Stereo Tracking: The local frame of the region that we want to track is computed using least square plane fitting over point cloud data in the ROI, where the 3D point cloud is generated using an Intel RealSense 2. The location and orientation of this estimated ROI plane is fed into the visual servoing controller,

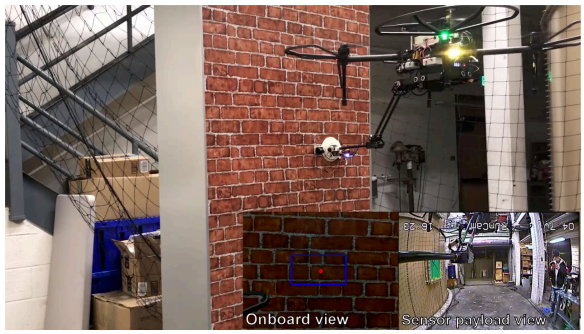

Fig. 9. A sensor placement trial. Inset are the image from the onboard camera tracking the ROI and the image from the payload camera.

which drives the quad to pre-determined poses relative to the planar surface.

2) State Machine: The state machine for this task is similar to the pick-and-place state machine, described in §VI-A, with an additional state for checking the adhesion of the payload. While planting the payload on the wall, the state machine estimates the contact force between the arm and the wall. When pressing in, this estimate is negative, and when it falls below an experimentally determined threshold, the state machine transitions from the "Placing" state into the new "Checking" state. The "Checking" state commands the robot to pull away slightly from the ROI. If the payload has adhered to the surface, the estimated contact force will become positive. When it exceeds another threshold, the gripper releases the payload and the robot retreats to a safe distance. If the force does not exceed the threshold in a configured time interval, the placement is deemed a failure, and the robot resets and tries again to place the sensor.

3) Results: The sensor placement application is demonstrated in the media attachements included with this work. An example frame from this video is shown in Figure 9. Three videos were taken of the experiments: one from an external vantage point, one from the onboard camera demonstrating the ROI tracking, and one from the payload camera. Figure 4 shows the estimated external force over the course of the trial.

VII. CONCLUSION

This work developed an aerial manipulation system using a commercial quadrotor, a custom arm and end-effector, and a new software framework for aerial autonomy capable of fault-tolerant industrial pick-and-place and remote sensor placement tasks. While failure detection and system health monitoring increased the robustness of the system, more robust hardware and environment-adaptive manipulation are necessary to further reduce the failure modes and drive the system toward 100% reliability. Future work will integrate advanced adaptive models for the quadrotor and the arm that explicitly take into account their coupled dynamics in order to reduce position control error in MPC methods. Finally, while we were able to demonstrate reliable and relatively efficient operation, the overall speed and agility of the robot can be further improved through improved modeling of the system dynamics and having access to the

lower level controls of the quadrotor. Achieving extreme agility without sacrificing reliability remains a central challenge yet to be solved.

REFERENCES

[1] Amazon, "Prime air," 2017. [Online]. Available: https://www.amazon.com/Amazon-Prime-Air/b?node=8037720011

[2] N. Michael, J. Fink, and V. Kumar, "Cooperative manipulation and transportation with aerial robots," *Auton. Robots*, vol. 30, no. 1, pp. 73–86, 2011.

[3] T. Lima *et al.*, "Vision based target interception using aerial manipulation," 2020, *arXiv:2009.13066*.

[4] AeroWorks, 2017. [Online]. Available: http://www.aeroworks2020.eu/

[5] *Aeroarms*, 2017. [Online]. Available: https://aeroarms-project.eu/

[6] F. Ruggiero, V. Lippiello, and A. Ollero, "Aerial manipulation: A literature review," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 1957–1964, Jul. 2018.

[7] C. D. Bellicoso, L. R. Buonocore, V. Lippiello, and B. Siciliano, "Design, modeling and control of a 5-dof light-weight robot arm for aerial manipulation," in *Proc. IEEE 23rd Mediterranean Conf. Control Automat.*, 2015, pp. 853–858.

[8] S. B. Backus and A. M. Dollar, "A prismatic-revolute-revolute joint hand for grasping from unmanned aerial vehicles and other minimally constrained vehicles," *J. Mechanisms Robot.*, vol. 10, no. 2, 2018, Art. no. 0 25006.

[9] P. E. Pounds, D. R. Bersak, and A. M. Dollar, "The yale aerial manipulator: Grasping in flight," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 2974–2975.

[10] A. Suarez, G. Heredia, and A. Ollero, "Lightweight compliant arm with compliant finger for aerial manipulation and inspection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 4449–4454.

[11] A. Suarez, G. Heredia, and A. Ollero, "Design of an anthropomorphic, compliant, and lightweight dual arm for aerial manipulation," *IEEE Access*, vol. 6, pp. 29173–29189, 2018.

[12] A. Suarez, F. Real, V. M. Vega, G. Heredia, A. Rodriguez-Castaño, and A. Ollero, "Compliant bimanual aerial manipulation: Standard and long reach configurations," *IEEE Access*, vol. 8, pp. 88844–88865, 2020.

[13] K. Bodie *et al.*, "An omnidirectional aerial manipulation platform for contact-based inspection," in *Robot.: Sci. Syst. (RSS)*, vol. 15, 2019.

[14] D. Chaikalis, F. Khorrami, and A. Tzes, "Adaptive control approaches for an unmanned aerial manipulation system," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2020, pp. 498–503.

[15] D. Tzoumanikas, F. Graule, Q. Yan, D. Shah, M. Popovic, and S. Leutenegger, "Aerial manipulation using hybrid force and position nmpc applied to aerial writing," in *Robot.: Sci. Syst. (RSS)*, vol. 16, 2020.

[16] J. Lee *et al.*, "Visual-inertial telepresence for aerial manipulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 1222–1229.

[17] A. Suarez, V. M. Vega, M. Fernandez, G. Heredia, and A. Ollero, "Benchmarks for aerial manipulation," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 2650–2657, Apr. 2020.

[18] A. Gawel *et al.*, "Aerial picking and delivery of magnetic objects with mavs," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 5746–5752.

[19] M. Nieuwenhuisen *et al.*, "Collaborative object picking and delivery with a team of micro aerial vehicles at mbzirc," in *Proc. IEEE Eur. Conf. Mobile Robots*, 2017, pp. 1–6.

[20] H. Lee, H. Kim, W. Kim, and H. J. Kim, "An integrated framework for cooperative aerial manipulators in unknown environments," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 2307–2314, Jul. 2018.

[21] D. Kim and P. Y. Oh, "Lab automation drones for mobile manipulation in high throughput systems," in *Proc. IEEE Int. Conf. Consum. Electron.*, 2018, pp. 1–5.

[22] M. Orsag, C. Korpela, S. Bogdan, and P. Oh, "Dexterous aerial robots-mobile manipulation using unmanned aerial systems," *IEEE Trans. Robot.*, vol. 33, no. 6, pp. 1453–1466, Dec. 2017.

[23] M. Perez-Jimenez, P. Ramon-Soria, B. Arrue, and A. Ollero, "Hecatonquiros: Open-source hardware for aerial manipulation applications," *Int. J. Adv. Robotic Syst.*, vol. 17, no. 2, 2020, Art. no. 1729881420921622.

[24] T. Lee, M. Leoky, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *Proc. 49th IEEE Conf. Decis. Control*, 2010, pp. 5420–5425.

[25] G. Garimella and M. Kobilarov, "Towards model-predictive control for aerial pick-and-place," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 4692–4697.

[26] G. Garimella, M. Sheckells, and M. Kobilarov, "Robust obstacle avoidance for aerial platforms using adaptive model predictive control," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 5876–5882.

[27] G. Baraban, M. Sheckells, S. Kim, and M. Kobilarov, "Adaptive parameter estimation for aerial manipulation," in *Proc. Amer. Control Conf.*, 2020, pp. 614–619.

[28] V. Ghadiok, J. Goldin, and W. Ren, "Autonomous indoor aerial gripping using a quadrotor," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 4645–4651.

[29] R. R. Ma, L. U. Odhner, and A. M. Dollar, "A modular, open-source 3D printed underactuated hand," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013, pp. 2737–2743.

[30] *Alvar*, 2017. [Online]. Available: http://virtual.vtt.fi/virtual/proj2/multimedia/alvar/index.html