A Framework for Reliable Aerial Manipulation

Gowtham Garimella*[1], Matthew Sheckells*[2], Soowon Kim[1] and Marin Kobilarov[1]

***Abstract*— This paper describes the development of an industrial package sorting capability using aerial manipulators. Our goal is to enable the transport of small packages to and from locations that are difficult or dangerous to access or to simply replace tedious repetitive tasks that otherwise require human labor or large and expensive industrial robots. This work focuses on the development of a framework that can achieve a high level of reliability but also efficiency. To accomplish this, we developed a novel software framework with built-in robustness to algorithmic failures and hardware faults. The software framework provides a way to combine independent modular behaviors, such as waypoint tracking and visual servoing, into a set of connected domain-dependent state machines. Motion robustness is achieved through model-based control that can adaptively compensate for the additional torques, forces, and other biases from the interaction of the manipulator and packages, as well as for variations in battery power. To simplify object picking and reduce the need for extremely precise end-effector positioning, the aerial manipulator is endowed with a novel magnetic gripper which mates with a flat receptacle attached to objects. In addition, a novel grasping strategy is employed to improve the probability of picking objects. Combining the fault-tolerant state machine framework with adaptive controllers we demonstrate that package sorting can be done reliably (85% end-to-end transport success rate, with 20% of errors due to software and 80% due to hardware and sensor failure) while maintaining the agility of the aerial manipulation system (e.g., each box is detected and picked up in less than 12 seconds on average).**

I. Introduction

Vertical take-off and landing (VTOL) vehicles such as quadrotors have gained a lot of attention recently due to their agility and ability to navigated 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, and structural maintenance applications [3], [4]. Many of these applications require interactions between multiple software components and hardware subsystems while navigating cluttered environments to achieve a desired goal. There are also performance constraints on tasks which would require navigating through the environments quickly. In such a scenario, the safety and reliability of the overall system under software and hardware failures is critical. In particular, the task of aerial manipulation is non-trivial since it involves underactuated quadrotor systems combined with multi-degree of freedom manipulators interacting with the environment. We propose a two-fold approach: on the software side, a fault tolerant state machine framework that implements several controllers for aerial manipulation 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 with a high probability of picking objects (90%) and tolerance to a wide variety of errors.

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

A. Related Work

Past research has focused on developing control algorithms and manipulators specifically for aerial manipulation (e.g. [5], [6], [7]). While results have been reported separately for various aspects of aerial manipulation such as control algorithms (e.g. [8], [9], [10], [11], [12]), motion planning, and visual servoing (e.g. [13], [14], [15]), very few fully-integrated systems that allow the combination of these basic behaviors into complex tasks with fault-recovery have been reported. Current commercially available aerial autonomy suites [16], [17] are limited to basic navigation and observation tasks and not directly applicable to aerial manipulation.

A few fully integrated applications for aerial manipulation

[1]Gowtham Garimella Soowon Kim and Marin Kobilarov are with the Department of Mechanical Engineering, Johns Hopkins University, 3400 N Charles Str, Baltimore, MD 21218, USA ggarime1|skim386|marin@jhu.edu

[2]Matthew Sheckells is with the Department of Computer Science, Johns Hopkins University, 3400 N Charles Str, Baltimore, MD 21218, USA msheckells@jhu.edu

[3]https://github.com/jhu-asco/aerial_autonomy

* These authors contributed equally.

have been proposed in recent years. An aerial manipulation 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.

This work proposes a reliable aerial manipulation system, at the core of which lies an autonomy software framework that is robust to controller and hardware failures. We apply the state machine framework to a package sorting application that combines an off-the-shelf quadrotor with a custom built light-weight 2-DoF arm and a magnetic gripper that is tolerant to position error. We implement and compare two different control strategies for picking objects: a PID controller that assumes tight inner-loop attitude control and a Model Predictive Controller (MPC). A novel magnetic gripper is developed that can grasp objects with a tolerance of 2 cm in end-effector position. We tested the entire system through a set of 101 experiments and documented different failure modes that can occur. The software framework is robust enough to complete 85 out of the 101 pick-place trials conducted. Finally, we provide the state machine framework and aerial manipulation controllers as open-source software[3].

II. Software Framework

At the core of this system lies a software framework that allows users to easily create autonomy applications by combining modular state behaviors, controllers, and hardware capabilities into domain-specific state machines. The software framework has been designed to:

- combine modular behaviors, such as waypoint navigation and visual servoing, into complex state machines to perform complicated tasks, like object pick-and-place;
- enable robustness to sensor, controller, and hardware failure, through introspection and fail-safe actions;
- provide control methods that adapt to environment changes;
- provide automated 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 software framework consists of two major components– the state machine and the robot system, which are described next.

A. From behaviors to automatically generated state machine

The state machine and its set of behaviors form the core of our software framework. The state machine logic is specified through a state transition table which consists of a list of tuples that specify the start state, transition event, ending state, transition action, and transition guard.

The states in a state machine denote the different stages during the execution of a task. In the context of aerial manipulation, the states denote different stages of the pick and place task. For example, the “Reaching Goal” state refers to when the quadrotor is navigating to a goal location. Similarly, the “Taking Off” state denotes the quadrotor in the process of taking off from the ground.

State transitions are triggered by an event. For example, a transition from the “Landed” state to the “Taking Off” state is triggered by a ”Take-off” event. Events are typically generated by the state machine itself or by users through a graphical interface.

When a state transition is triggered, a guard function first verifies the feasibility of a transition between two states. That is, a state transition occurs only when the guard allows it. If the guard blocks a state transition, the current state will remain unchanged. As an example, in our pick-and-place task, a guard function checks that the output of the object tracking module is valid before transitioning into a “Visual Servoing” state, which attempts to align the quadrotor with an object using visual features.

When the guard allows a state transition to occur, an associated transition action executes. Typically, these actions switch the active controllers, send direct commands to the hardware driver, or configure some aspect of the robot for the new state. The transition actions can be triggered by the user or through state machine logic. These actions can be chained to create more complicated actions and thereby reduce code duplication.

While in a particular state, the system repeatedly executes an internal behavior associated with that state. These behaviors, called “internal actions”, trigger specific events on the state machine based on the current robot state. These actions typically perform health checks on the hardware and controllers and check for convergence of active controllers. In the case of the “Reaching Goal” state referred to above, the internal action checks for the battery status of the quadrotor and triggers an “Abort” event if the battery is low and checks if the quadrotor has converged to the goal location, triggering a “Completed” event if it has converged. Similar to transition actions, the internal actions can also be chained together. Encoding the state machine logic in the internal actions allows for decentralization of state machine logic and reduces code duplication among different state machines.

The state machine has been implemented using the Boost meta state machine library in C++ [23]. The states, actions, and guards in the table are C++ classes that can be reused

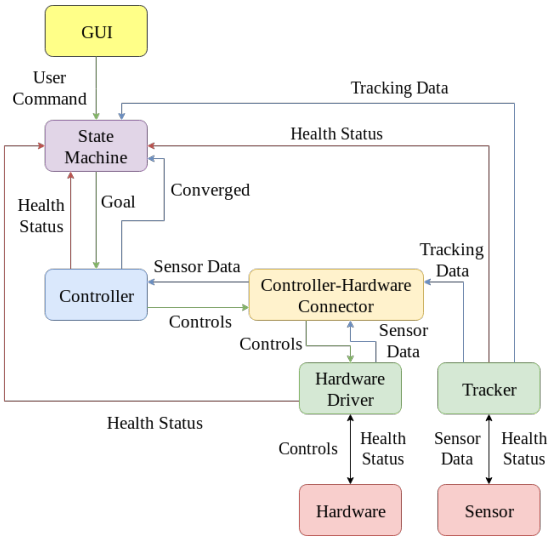

Fig. 2. Illustration of the interaction between the various software components of the developed framework. The Graphical User Interface (GUI) displays feedback from the individual components, but those connections are left out of the diagram for readability.

in different transition tables to form diverse state machine behaviors without code duplication.

B. Robot System

Figure 2 illustrates the interaction between different components of the robot system and the state machine. The data obtained from sensors and hardware drivers are fed into the state machine. The state machine then selects the active set of controllers and the desired goal for these controllers. The controllers are further grouped based on the hardware type that they command so that only a single controller is allowed to execute for a specific hardware group. Choosing a single controller avoids accidental errors, such as when two types of position controllers are trying to send conflicting commands to the quadrotor. The output of the controllers is finally sent back to the physical robot through the hardware driver.

C. Graphical User Interface

We have also developed a Graphical User Interface (GUI) from which users can trigger events and monitor the robot system and state machine health. The GUI communicates with the state machine through the Robot Operating System (ROS) middleware [24] (Figure 3).

III. Aerial Manipulator Control

We now describe in detail 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

Many off-the-shelf quadrotors come equipped with built-in flight control hardware, where the control interface is limited to higher level commands such as roll-pitch-yaw-thrust or angular rate commands instead of direct motor commands.

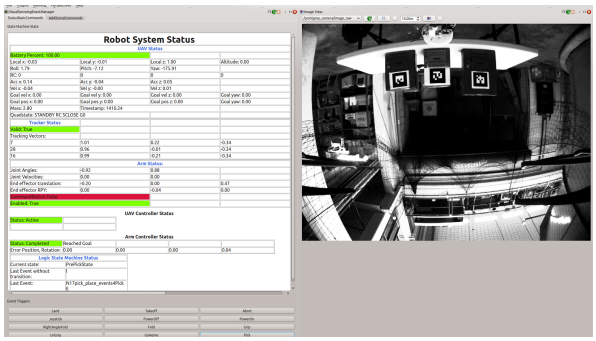

Fig. 3. User interface for sending commands and viewing state machine and robot system feedback. The left panel is the portion provided by the framework showing the state machine status. The right panel shows a view from the onboard camera with objects to pick.

Here, we describe a controller that can be employed on such systems.

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 attitude, $v \in \mathbb{R}^3$ 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

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

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. The proportional and derivative gains for the x and y axes are selected separately from that of the z-axis gain since the quadrotor dynamics are significantly different along the z-axis.

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, 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 since 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

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

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

The above conversion has a singularity as 90° degrees roll, which is not encountered in our application.

The commanded yaw rate is proportional to the error between the current yaw and desired yaw obtained from the reference trajectory as

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

Previous work proves stability for a similar class of trajectory tracking controllers that use PID to compute a desired force and an inner-loop attitude controller to achieve the desired force direction [25].

B. Model Predictive Controller

The model predictive controller computes the thrust and desired attitude for the quadrotor by solving a trajectory optimization problem. The optimization minimizes the cost over a predicted trajectory for the quadrotor using a sequence of control inputs. In this work, we assume a second order model of the quadrotor rotational dynamics as explained in [26]. The arm is assumed to be a kinematic system and is controlled separately. The state of the quadrotor for MPC optimization is obtained by extending the regular quadrotor state by the desired Euler angles ξ_d so that $x = (p, R, v, w, \xi_d)$. The control inputs for the system are given by the thrust u_t and the Euler angle rates $\dot{\xi}_d$. The dynamics model as described in [26] predicts the state at step $i{+}1$ given the state x_i and control u_i and any additional parameters p_i at step i as

$$x_{i+1} = f(x_i, u_i, p_i). \quad (7)$$

In our model, the external disturbances (modeled as accelerations) and thrust gain parameters are taken as parameters to the system.

The cost function used for the optimization is a quadratic cost function that minimizes the error between quadrotor state x_i and reference state $\bar{x}_i$ while also minimizing the deviation of the control effort u_i from the desired control effort $\bar{u}_i$. The cost function can be written as

$$L = (x_N - \bar{x}_N)^\top Q_N (x_N - \bar{x}_N) + \sum_{i=0}^{N-1} (x_i - \bar{x}_i)^\top Q (x_i - \bar{x}_i) + (u_i - \bar{u}_i)^\top R (u_i - \bar{u}_i), \quad (8)$$

where N is the number of trajectory steps, Q_N is the terminal cost gain, and Q, R are the cost gains along the trajectory. The MPC trajectory optimization minimizes the cost function L subject to the constraint that the trajectory is dynamically feasible

$$u^*_{1:N} = \underset{u_{1:N}}{\arg\min}\, L \ \text{ s.t } x_{i+1} = f(x_i, u_i, p_i), \quad (9)$$

giving the optimal control inputs $u^*_{1:N}$. The trajectory optimization problem is inherently sparse since the controls at stage i can only effect the states $i+1$ to N. Thus, we use a Stagewise Newton method [27] which is also closely related to Differential Dynamic Programming (DDP) [28] to solve (9). During a single iteration, this method performs a backward pass over the current control trajectory which computes the gradient of L with respect to $u_{1:N}$ and executes a forward pass which performs a line search over the descent step size by fully unrolling the dynamics to ensure the cost decreases. We employ the Casadi automatic differentiation library [29] to find the gradients of the dynamics required for the Stagewise Newton method. The MPC optimization for the quadrotor dynamics is able to run at a frequency of 100 Hz on an onboard Intel NUC i5 computer.

C. Reference Trajectory Generation

The trajectory tracking controllers described above need a reference trajectory that is feasible for the quadrotor to track. When navigating to a waypoint or approaching an object to pick it up, 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 are given 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.

1) 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. This results in a periodic "poking" motion. This behavior is desirable since it 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.

IV. Parameter Estimation

A. Thrust Gain Estimation

The acceleration-based controller relies on the autopilot to achieve the desired thrust, roll, pitch, and yaw rate. The autopilot usually takes as input a normalized thrust command between 0 and 100, where a non-constant scale factor can transform the normalized value to a metric unit of thrust force. The scale factor, called the thrust gain, is constantly changing since it depends on the battery voltage and mass of the quadrotor. To compensate for these effects, a thrust gain estimator computes the mapping between the thrust command and the actual thrust force based on the commanded thrust, the body acceleration vector, and the orientation obtained from the quadrotor. 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

$$a = k_t R e_3 u + g \tag{10}$$

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

$$k_t = \frac{1}{u} e_3^T (a_b - R^\top g) \tag{11}$$

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

$$\bar{k}_{t_{i+1}} = (1 - \lambda)\bar{k}_{t_i} + \lambda k_{t_i}, \tag{12}$$

where $\bar{k}_{t_i}$ is the filtered thrust gain estimate at time index i. By choosing a scale λ between 0 and 1, the thrust gain can be adjusted to change more aggressively, which leads to the quadrotor changing thrust aggressively to compensate for a change in mass. Figure 4 shows the thrust gain estimated for the quadrotor during a pick-and-place application. The positive jumps in the gain denote a package being dropped and a negative jump denotes a package being picked. The thrust gain exhibits an overall downward trend as the battery voltage drops over time.

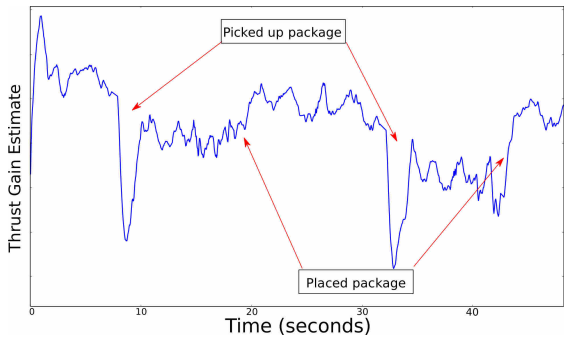

Fig. 4. Estimate of thrust gain k_t computed from IMU data and expected acceleration.

B. Euler Angle Bias Estimation

We also found a small difference of approximately 0.5 degrees between the roll and pitch Euler angles reported by the IMU and the angles obtained by inverting the fused body acceleration reported by the IMU a_{acc} as shown in Figure 5. The roll and pitch angles corresponding to fused body acceleration are obtained using (5) where desired a_d is replaced by the rotated body acceleration reported by the IMU, that is

$$a_{global} = R_Y(\theta) R_X(\phi) a_{acc}, \tag{13}$$

where $\bar{a}_{global} = (a_{global} + g)/\|a_{global} + g\|$. 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 commanded roll and pitch before sending them to the autopilot, where

$$\delta_\phi = \phi - \phi_{acc}, \quad \delta_\theta = \theta - \theta_{acc}. \tag{14}$$

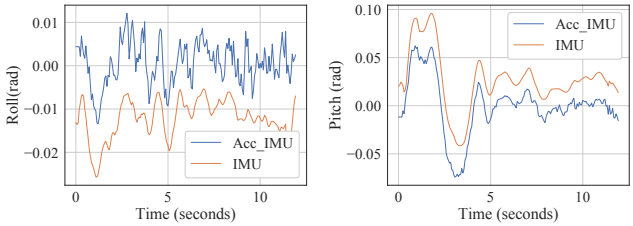

Fig. 5. Bias in roll and pitch estimated from difference in expected and actual accelerations.

V. Hardware

A. Commercial Off-the-Shelf quadrotor

The aerial manipulation system contains 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 over a UART connection.

B. Motion capture system

Motion capture system is used for position and velocity estimation for the control algorithms. The DJI Guidance sensor suite consisting of 5 stereo cameras is used as a fail-safe in case we lose motion capture during experiments. Although using a motion capture system is restricting the applicability of the system, it provides a good ground truth for comparing our system with other future applications.

C. Manipulator

1) Custom 2-DoF Arm: Several previous works, like [30] and [5], 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. Since 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. Since 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 OpenHand [31], are too heavy and do not fit the requirements specified above. Our custom gripper shown in Figure 6 is composed of four magnets with alternating polarity embedded into a wheel attached to a servo. The magnets are attracted to a mating

joint (shown in Figure 6) that is attached to any object the user wishes to pick. The mating joint has a pattern of magnets to give the gripper more than one place to attach, thereby increasing the amount of position error it can tolerate while picking. It can tolerate a position error of about 3cm parallel to the surface of the plate and 2cm perpendicular to the plate. 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. An onboard Teensy microcontroller runs software which sends commands to the servo and receives feedback from the switch. The gripper communication channel connects to a single servo communication bus that runs up the length of the manipulator to the computer.

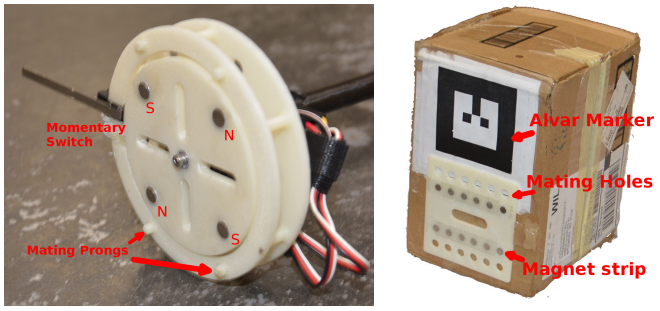

Fig. 6. The magnetic gripper (left) and a sample package (right) used in our aerial manipulation experiments. The package is instrumented with an AR marker to facilitate tracking and a magnetic mating joint so it can attach to the gripper.

VI. Industrial Pick-and-Place Application

The software framework developed in §II is used to develop an industrial pick-and-place application leveraging the aerial manipulation platform described in §V.

A. Setup

The goal of the application is to sort packages from a packaging area (table) and transport them to corresponding storage area (shelf). The package transportation capability can be useful, for instance, in package fulfillment centers or for remote object transport in radio-active environments. Overall, the application demonstrates the system's reliable aerial grasping and object insertion capabilities.

The packages are tagged with AR markers [32] and have an attached mating joint that connects to the gripper described in §V-C.2. 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).

1) State Machine: Figure 8 shows a simplified illustration of the finite state machine for the pick place application. There are two different logical loops in the diagram.

The first is the regular logic loop starting from "Waiting to Pick" state. During this cycle, the quadrotor 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.

Various system components could fail throughout the pick-and-place process, but the implemented state machine accounts for such failures through a second loop known as the fault-recovery loop. For example, during picking, the arm could block the marker from the camera, resulting in a tracking loss. Instead of just aborting and waiting for human input, the system instead back-tracks to its prior position and re-attempts the picking process. Other failure modes include failing to pick the object within a specified timeout. Recovery state transitions are shown in red in Figure 8.

In addition to automatic recovery, sometimes during the experiment, a user intervention is necessary. A list of such failures encountered during operations is listed in Table II. The state machine ensures the system is safe under these failure modes by switching to hovering and relying on internal controller to stably hover in place until the user is ready to intervene. Furthermore, the state machine accepts manual override from a safety pilot to abort any action safely. The state machine recognizes the intervention and aborts any active controllers running on the machine. Therefore, the user can resume picking operation after rectifying the error and disabling the override sent.

B. Results

Figure 7 shows a timeline 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.

We quantified the ability of the quadrotor to perform a successful pick operation over 101 trials of picking and placing. 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 9 compares the mean absolute errors along translational positions, velocities, and yaw angle for each controller. Both the MPC controller and acceleration-based controller performed well during trajectory tracking, but the acceleration-based controller with more extensive gain tuning produced slightly better results.

Table I shows the mean trajectory tracking errors and pick times during the trials. The aerial manipulator was able to pick the object successfully 80% of the time without the ability to detect system faults. The system's pick success

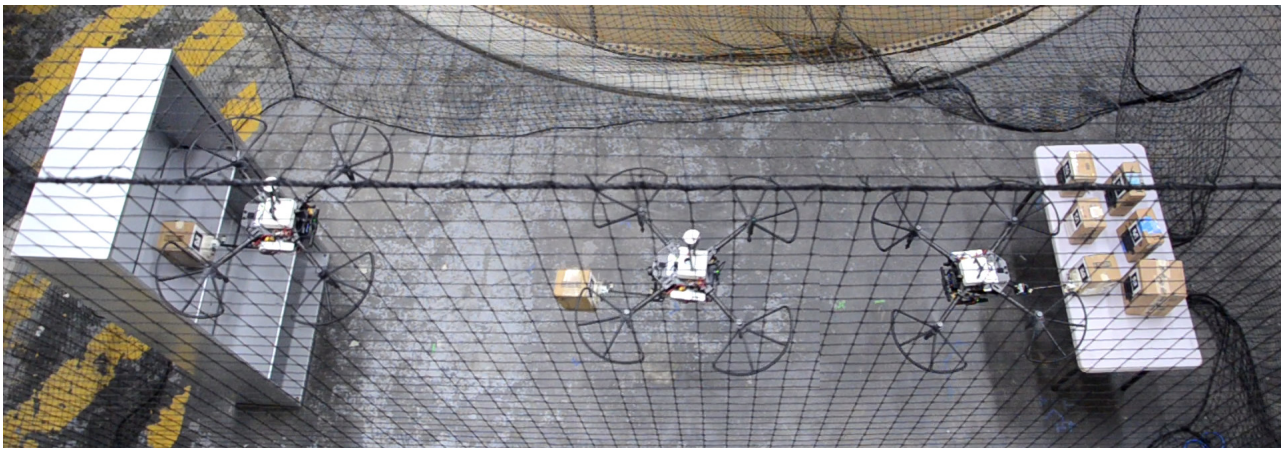

Fig. 7. Time-lapse of the aerial manipulator picking the object highlighted in yellow from the packaging area (top-left) and placing it on a storage shelf (top-right) in the same trial. The bottom picture shows an overhead view of the pick-and-place procedure.

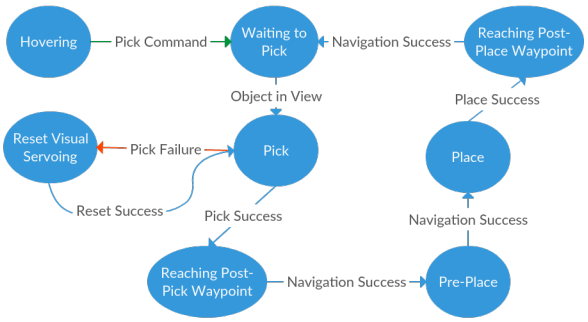

Fig. 8. Part of the state machine for picking and placing a package. The recovery actions are red and user actions are green. The user can also abort from any other state back to hovering if manual intervention is desired.

Pick Success Rate	91/101
Overall Success Rate	85/101
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

TABLE I

PICK SUCCESS RATE, PICK TIME STATISTICS, AND ERROR IN QUADROTOR POSITION AND YAW FOR PICKING AN OBJECT

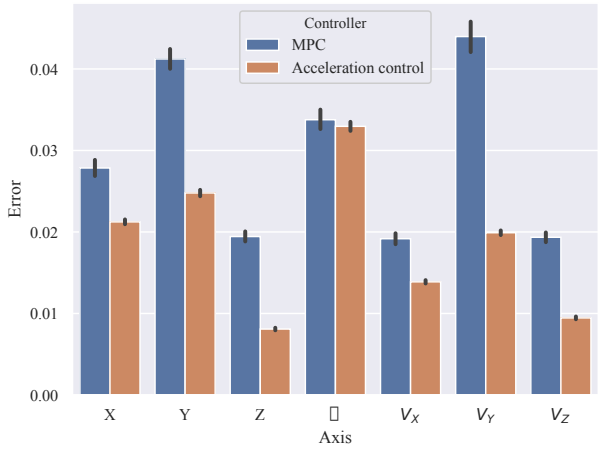

Fig. 9. Mean absolute errors along x, y, z (meters), and yaw ψ axes (radians) and translational velocities (meters/second) for MPC and acceleration-based controller. The black lines show the 95% confidence interval obtained using bootstrapping.

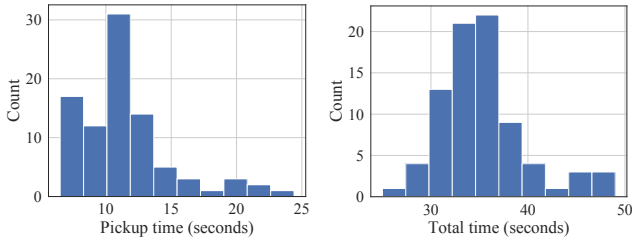

Fig. 10. Histogram of pickup and total time for placing one box using the pick place state machine.

rate increased to 85% when it is was able to automatically recognize failure to pick an object and could retry and re-pick the object in a future attempt. We also achieved a mean absolute error of less than 3 centimeters in all translational axis and less than 2 centimeters/second in velocity. Figure 10 shows a histogram of pickup times and total time for pick and place of an object over different trials. The majority of pickup times vary from 6 seconds to 16 seconds, while the total pick-and-place time for one box varies from 30 seconds to 40 seconds in most cases.

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 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 Mode	Number of Failures
Object misplaced in shelf while placing	1
Gripper failed to hold onto object	1
Lost motion capture while gripping	1
Controller failed after multiple retries	3
Proximity sensor failed to detect object	3
Object went out of workspace	3
Camera stops responding due to driver errors	3

TABLE II

FAILURE MODES DURING PICK-AND-PLACE TRIALS

failure modes shown in Table II 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. New grippers that do not require custom attachments on the package will also be designed to make the system more widely applicable. Finally, while we were able to demonstrate reliable and relatively efficient operation, the overall speed and agility of the robot can be further improved. Achieving extreme agility without sacrificing reliability remains a central challenge yet to be solved.

VIII. ACKNOWLEDGEMENTS

Thanks to Professor Joseph Katz and Professor Louis Whitcomb for providing facilities for conducting experiments and to Subhransu Mishra for guidance in designing the manipulator hardware. This material is based upon work supported by the National Science Foundation under grant no:1527432.

REFERENCES

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

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

[3] AeroWorks. http://www.aeroworks2020.eu/, 2017.

[4] AEROARMS. https://aeroarms-project.eu/, 2017.

[5] 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 *Control and Automation (MED), 2015 23th Mediterranean Conference on*, pp. 853–858, IEEE, 2015.

[6] A. Suarez, G. Heredia, and A. Ollero, "Lightweight compliant arm with compliant finger for aerial manipulation and inspection," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4449–4454, Oct 2016.

[7] P. E. Pounds, D. R. Bersak, and A. M. Dollar, "The yale aerial manipulator: grasping in flight," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2974–2975, IEEE, 2011.

[8] R. Mebarki and V. Lippiello, "Imagebased control for aerial manipulation," *Asian Journal of Control*, vol. 16, no. 3, pp. 646–656, 2014.

[9] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar, "Design, modeling, estimation and control for aerial grasping and manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2668–2673, IEEE, 2011.

[10] K. Kondak, F. Huber, M. Schwarzbach, M. Laiacker, D. Sommer, M. Bejar, and A. Ollero, "Aerial manipulation robot composed of an autonomous helicopter and a 7 degrees of freedom industrial manipulator," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2107–2112, IEEE, 2014.

[11] C. Korpela, P. Brahmbhatt, M. Orsag, and P. Oh, "Towards the realization of mobile manipulating unmanned aerial vehicles (mm-uav): Peg-in-hole insertion tasks," in *International Conference on Technologies for Practical Robot Applications (TePRA)*, pp. 1–6, IEEE, 2013.

[12] S. Kim, S. Choi, and H. J. Kim, "Aerial manipulation using a quadrotor with a two dof robotic arm," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4990–4995, IEEE, 2013.

[13] K. Kondak, F. Huber, M. Schwarzbach, M. Laiacker, D. Sommer, M. Bejar, and A. Ollero, "Aerial manipulation robot composed of an autonomous helicopter and a 7 degrees of freedom industrial manipulator," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2107–2112, May 2014.

[14] V. Lippiello, J. Cacace, A. Santamaria-Navarro, J. Andrade-Cetto, M. A. Trujillo, Y. R. Esteves, and A. Viguria, "Hybrid visual servoing with hierarchical task composition for aerial manipulation," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 259–266, 2016.

[15] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, *Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera*, pp. 235–252. Cham: Springer International Publishing, 2017.

[16] "Dji guidance pro suite." https://www.dji.com/ground-station-pro, 2017.

[17] L. Meier, "Pixhawk autopilot - px4 autopilot platform," 2014.

[18] A. Gawel, M. Kamel, T. Novkovic, J. Widauer, D. Schindler, B. P. von Altishofen, R. Siegwart, and J. Nieto, "Aerial picking and delivery of magnetic objects with mavs," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5746–5752, IEEE, 2017.

[19] M. Nieuwenhuisen, M. Beul, R. A. Rosu, J. Quenzel, D. Pavlichenko, S. Houben, and S. Behnke, "Collaborative object picking and delivery with a team of micro aerial vehicles at mbzirc," in *European Conference on Mobile Robots (ECMR)*, pp. 1–6, IEEE, 2017.

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

[21] D. Kim and P. Y. Oh, "Lab automation drones for mobile manipulation in high throughput systems," in *IEEE International Conference on Consumer Electronics (ICCE)*, pp. 1–5, IEEE, 2018.

[22] M. Orsag, C. Korpela, S. Bogdan, and P. Oh, "Dexterous aerial robotsmobile manipulation using unmanned aerial systems," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1453–1466, 2017.

[23] "Boost meta state machine msm library," 2017.

[24] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, 2009.

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

[26] G. Garimella, M. Sheckells, and M. Kobilarov, "Robust obstacle avoidance for aerial platforms using adaptive model predictive control," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5876–5882, IEEE, 2017.

[27] D. P. Bertsekas, *Nonlinear Programming, 2nd ed.* Belmont, MA: Athena Scientific, 2003.

[28] D. H. Jacobson and D. Q. Mayne, *Differential dynamic programming*. Modern analytic and computational methods in science and mathematics, New York: Elsevier, 1970.

[29] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, In Press, 2018.

[30] V. Ghadiok, J. Goldin, and W. Ren, "Autonomous indoor aerial gripping using a quadrotor," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4645–4651, IEEE, 2011.

[31] R. R. Ma, L. U. Odhner, and A. M. Dollar, "A modular, open-source 3d printed underactuated hand," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2737–2743, IEEE, 2013.

[32] Alvar. http://virtual.vtt.fi/virtual/proj2/multimedia/alvar/index.html, 2017.