A Framework for Reliable Aerial Manipulation

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

***Abstract*— This paper describes the development of a practical aerial manipulation system. Our goal is to enable the transport of small payloads to and from locations that are difficult or dangerous to access and replace tedious repetitive tasks that 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 objects. To have efficient end-effector positioning, the aerial manipulator is endowed with a novel magnetic gripper which mates with a flat receptacle attached to objects. Combining the fault-tolerant state machine framework with adaptive controllers we demonstrate that aerial package sorting can be done reliably (85% end-to-end transport success rate) 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). Furthermore, we leverage the same set of modular behaviors to perform a remote sensor placement task, where the robot successfully attaches an adhesive payload to a remote region-of-interest.**

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, and structural maintenance applications [3], [4]. Many of these applications require interactions between multiple software components and hardware subsystems while navigating complex environments to achieve a desired goal. In such scenarios, 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 as it involves underactuated quadrotor systems combined with multi-degree of freedom manipulators interacting with the environment.

[1]G. Garimella, S. Kim, G. Baraban, and M. Kobilarov are with the Department of Mechanical Engineering, Johns Hopkins University, 3400 N Charles Str, Baltimore, MD 21218, USA ggarime1|skim386|gbaraba1|marin@jhu.edu

[2]M. 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.

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

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 that is demonstrated on a pick-and-place scenario and remote sensor payload placement task.

A. Related Work

Past research has focused on developing control algorithms and manipulators specifically for aerial manipulation (e.g. [5], [6], [7], [8], [9]). While results have been reported separately for various aspects of aerial manipulation such as control algorithms (e.g. [10], [11], [12], [13], [14]), motion planning, and visual servoing (e.g. [15], [16], [17]), 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 [18], [19] are limited to basic navigation and observation tasks and not directly applicable to aerial manipulation.

A few fully integrated applications for aerial manipulation have been proposed in recent years. An aerial manipulation system for moving metallic discs and sheets is proposed by [20], [21]. The system developed by Gawel et al. [20] 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

Manuscript 1644 submitted to 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 1, 2019.

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 [22]. Kim et al. developed an aerial manipulation system for lab automation using a parallel manipulator [23]. Orsag et al. suggested a benchmark for different aerial grasping applications [24]. 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.

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 two different tasks and documented various failure modes that can occur. 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 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 control methods that adapt to environment changes; provide automated tests for controllers and logic systems, independent of their hardware implementation; and 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.

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 attitude, $v \in \mathbb{R}^3$ is the velocity, and $\omega \in \mathbb{R}^3$ is the angular velocity. The

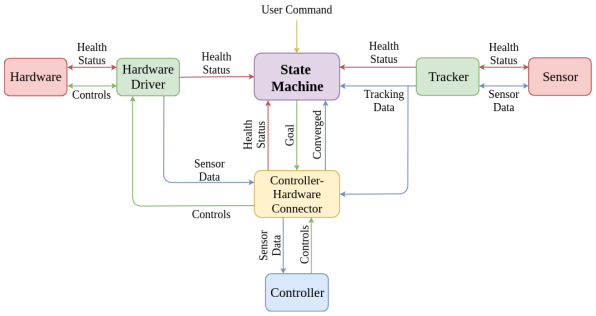

Fig. 2. Illustration of the interaction between the various software components of the developed framework.

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

Manuscript 1644 submitted to 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 1, 2019.

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 PID to compute a desired force and an inner-loop attitude controller to achieve the desired force direction [25].

The arm is assumed to be a kinematic system and is controlled independently of the quadrotor.

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

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

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

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

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

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

giving the optimal control inputs $u^*_{1:N}$. The trajectory optimization problem is inherently sparse as 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). 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

Two strategies are used to generate reference trajectories for navitagion 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.

IV. Online System Identification

A. Thrust Gain Estimation

The acceleration-based controller relies on the autopilot to achieve the desired thrust, roll, pitch, and yaw rate. The autopilot 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 as 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 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.

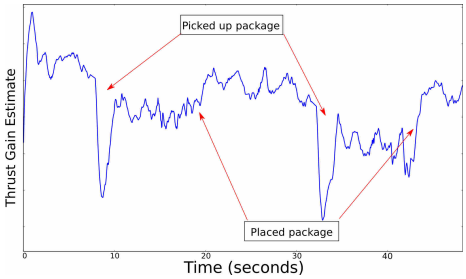

Fig. 3. Estimate of thrust gain k_t computed from IMU data and expected acceleration during pick-and-place task.

B. Euler Angle Bias Estimation

We also 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, that is

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

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

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

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

$$a_{bias} = Ra_b - k_t Re_3 u - g. \tag{15}$$

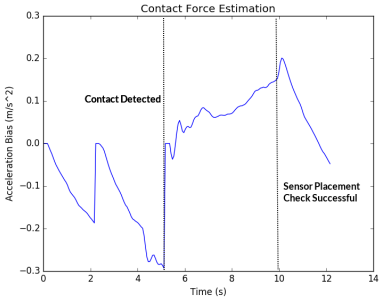

Fig. 4. The acceleration bias estimate during a sensor placement trial. The system uses a threshold on the contact force to determine when the payload has been pushed against the target surface. Once firmly pressed against the surface and before releasing the payload, the robot pulls on the payload to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

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.

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 [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. 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 OpenHand [31], 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

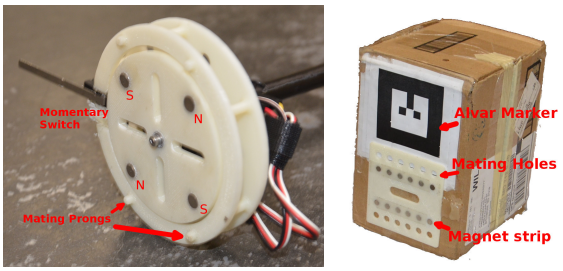

Fig. 5. The magnetic gripper (left) and a sample package (right) used in our aerial manipulation experiments. The package is instrumented with an AR marker to facilitate tracking and a magnetic mating joint so it can attach to the gripper.

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.

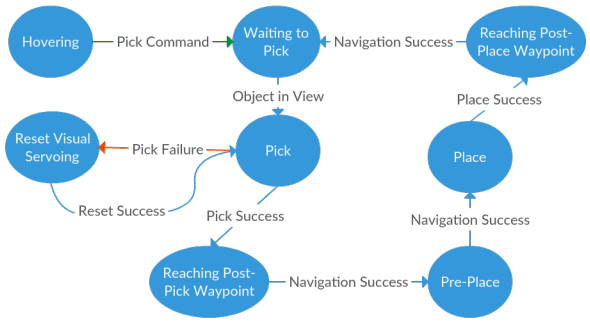

Fig. 6. Part of the state machine for picking and placing a package. The recovery actions are red and user actions are green. The user can also abort from any other state back to hovering if manual intervention is desired.

VI. Industrial Pick-and-Place Application

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

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) to demonstrate 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-B.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 6 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, is the camera loses track of the marker, 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 6.

The state machine ensures the system is safe under various failure modes by switching to hovering and relying on internal controller to stably hover in place until the user is ready to intervene.

B. Results

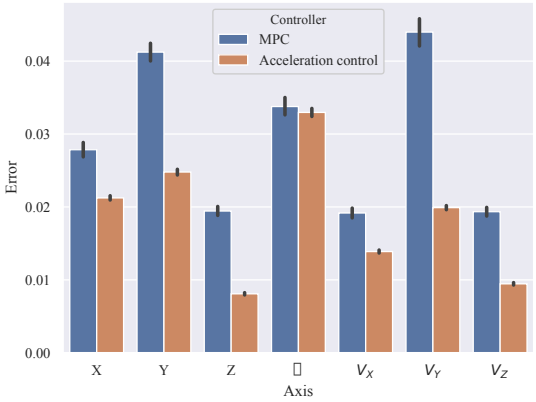

Fig. 8. Mean absolute errors along x, y, z (meters), and yaw ψ axes (radians) and translational velocities (meters/second) for MPC and acceleration-based controller. The black lines show the 95% confidence interval obtained using bootstrapping.

Figure 7 shows a timelapse of the pick-and-place task, where the quadrotor picks up a package from the table

Pick Success Rate	91/101
End-to-end 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-AND-PLACE TASK STATISTICS

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 8 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 rate increased to 90% 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 cm in all translational axis and less than 2 cm/s in velocity.

VII. REMOTE SENSOR PAYLOAD PLACEMENT

Re-using many of the same behaviors from §VI, 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.

A. Setup

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, 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, 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.

B. 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.

VIII. 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. Achieving extreme agility without sacrificing reliability remains a central challenge yet to be solved.

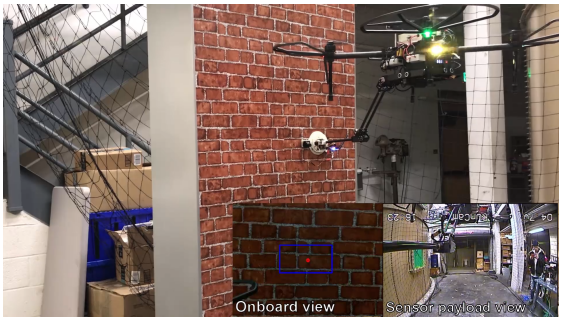

Fig. 9. A sensor placement trial. Inset are the image from the onboard camera tracking the ROI and the image from the payload camera.

Manuscript 1644 submitted to 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 1, 2019.

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] S. B. Backus and A. M. Dollar, "A prismatic-revolute-revolute joint hand for grasping from unmanned aerial vehicles and other minimally constrained vehicles," *Journal of Mechanisms and Robotics*, vol. 10, no. 2, p. 025006, 2018.

[8] 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.

[9] P. E. Pounds, D. R. Bersak, and A. M. Dollar, "Grasping from the air: Hovering capture and load stability," in *IEEE international conference on robotics and automation (ICRA)*, pp. 2491–2498, IEEE, 2011.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[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.